Mälardalen University Doctoral Dissertation 420

Scheduling and Dispatching Strategies for Real-Time Applications in Multi-Server Systems

Mohammed Salman Shaik

Mälardalen University

Mälardalen University Press Dissertations
No. 420

SCHEDULING AND DISPATCHING STRATEGIES FOR REAL-TIME APPLICATIONS IN MULTI-SERVER SYSTEMS

Mohammed Salman Shaik

2024

School of Innovation, Design and Engineering

ISBN 978-91-7485-683-5
ISSN 1651-4238
Printed by E-Print AB, Stockholm, Sweden

Mälardalen University Press Dissertations
No. 420

SCHEDULING AND DISPATCHING STRATEGIES FOR REAL-TIME APPLICATIONS IN MULTI-SERVER SYSTEMS

Mohammed Salman Shaik

Akademisk avhandling

som för avläggande av teknologie doktorsexamen i datavetenskap vid Akademin för innovation, design och teknik kommer att offentligen försvaras tisdagen den 5 november 2024, 13.15 i Kappa, Mälardalens universitet, Västerås.

Fakultetsopponent: Professor Marko Bertogna,
University of Modena and Reggio Emilia, Italy.

Akademin för innovation, design och teknik

Abstract

Real-time systems such as industrial robots and automated guided vehicles integrate a wide range of algorithms with varying levels of timing requirements to achieve their functional behavior. Historically, in certain systems, these algorithms were deployed on dedicated single-core hardware platforms that exchanged information over a real-time network, while more recent designs have adapted an integrated architecture where these algorithms are executed on an embedded multi-core hardware platform. The advantages provided by cloud and fog architectures for non-real-time applications have prompted discussions around the possibility of achieving similar advantages for systems such as industrial robot controllers by moving from an embedded architecture to a cloud and fog native architecture. This thesis addresses a subset of challenges related to scheduling to facilitate this transition and presents three main contributions aimed at improving online scheduling methodologies in multi-server systems for applications with real-time requirements. First, an approach based on minimum parallelism reservations is proposed for scheduling sequential tasks in hierarchical multi-server systems with clairvoyant inputs, ensuring adherence to hard real-time requirements. Second, a framework is introduced that utilizes estimated processing times to enhance average throughput in distributed multi-queue multi-server systems while managing tasks with stochastic inputs and firm real-time requirements, thereby improving resource utilization. Finally, competitive algorithms are proposed that leverage estimated processing times to minimize average (modified) tardiness in centralized single-queue multi-server systems, addressing the scheduling of sequential tasks with arbitrary arrivals and soft real-time requirements. Collectively, these contributions establish a robust foundation for improving the performance of real-time systems operating in increasingly complex environments characterized by dynamic workloads and varying resource availability.

ISBN 978-91-7485-683-5
ISSN 1651-4238

Abstract

Real-time systems are central components in, e.g., industrial robots and automated guided vehicles, which integrate a wide range of algorithms with varying levels of timing requirements to achieve their functional behavior. Historically, in certain systems, these algorithms were deployed on dedicated single-core hardware platforms that exchanged information over a real-time network, while more recent designs have adapted an integrated architecture where these algorithms are executed on an embedded multi-core hardware platform. Meanwhile, the advantages provided by cloud and fog architectures for non-real-time applications have prompted discussions about the possibility of achieving similar benefits for systems such as industrial robot controllers. This thesis addresses a subset of challenges related to scheduling to facilitate this transition, and presents three main contributions aimed at improving online scheduling approaches in multi-server systems for applications with real-time requirements. First, an approach based on minimum parallelism reservations is proposed for scheduling sequential tasks in hierarchical multi-server systems with clairvoyant inputs, ensuring adherence to hard real-time requirements. Second, a framework is introduced that utilizes estimated processing times to enhance average throughput in distributed multi-queue multi-server systems while managing tasks with stochastic inputs and firm real-time requirements, thereby improving resource utilization. Finally, competitive algorithms are proposed that leverage estimated processing times to minimize average (modified) tardiness in centralized single-queue multi-server systems, addressing the scheduling of sequential tasks with arbitrary arrivals and soft real-time requirements. Collectively, these contributions establish a robust foundation for improving the

performance of real-time systems operating in increasingly complex environments characterized by dynamic workloads and varying resource availability.

Sammanfattning

Realtidssystem är centrala komponenter i till exempel industrirobotar och automatiserade självkörande fordon. I dessa system integrerar realtidssystem ett brett utbud av algoritmer med varierande nivåer av tidskrav för att uppnå sitt funktionella beteende. Historiskt sett har dessa algoritmer i vissa system distribuerats på dedikerade enkärniga hårdvaru plattformar, som utbytte information över ett realtids nätverk, medan nyare design har anpassat en integrerad arkitektur där dessa algoritmer körs på en inbäddad flerkärnig hårdvaruplattform. Fördelarna med nyare så kallade moln- och dimarkitekturer för andra typer av tillämpningar har föranlett diskussioner kring möjligheten att uppnå liknande fördelar för exempelvis styrsystem för industrirobotar, genom att gå från en inbäddad arkitektur till en molnbaserad arkitektur. Denna avhandling tar upp en del utmaningar relaterade till schemaläggning för att underlätta en sådan övergång, och presenterar tre huvudsakliga bidrag som syftar till att förbättra online schemaläggningsmetoder i multi-serversystem för applikationer med realtidskrav. För det första föreslås ett tillvägagångssätt baserat på minsta parallellitetsreservationer för att schemalägga sekventiella uppgifter i hierarkiska multiserversystem med klärvoajanta indata, vilket säkerställer att hårda realtidskrav följs. För det andra introduceras ett ramverk som använder beräknade behandlingstider för att förbättra den genomsnittliga genomströmningen i distribuerade multi-queue multi-server system samtidigt som uppgifter hanteras med stokastiska indata och fasta realtidskrav, och därigenom förbättra resursutnyttjandet. Slutligen föreslås konkurrenskraftiga algoritmer som utnyttjar uppskattade bearbetningstider för att minimera genomsnittlig (modifierad) försening i centraliserade enkelköiga multi-

serversystem, vilket tar itu med schemaläggning av sekventiella uppgifter med godtyckliga ankomster och mjuka realtidskrav. Tillsammans skapar dessa bidrag en robust grund för att förbättra prestandan och effektiviteten hos realtidssystem som arbetar i allt mer komplexa miljöer som kännetecknas av dynamiska arbetsbelastningar och varierande resurstillgänglighet.

Acknowledgment

I would like to thank my PhD advisors Thomas Nolte, Alessandro Papadopoulos and Saad Mubeen for their constant encouragement and positive feedback that made this endeavor rather fulfilling. I would also like to thank Anders Wall, Fredrik Hedenfalk and Daniel Eriksson, for their continuous support during my time at ABB. I am glad to have had insightful discussions with Per Willför and Mikael Norrlöf, which deepened my understanding of the challenges in software engineering for industrial robots. I would also like to appreciate the support I received from my amazing colleagues, Daniel Sehlberg, Nicklas Larsson, Kjetil Erga, Daniel Watson and Erik Dahlberg!

I cherish the memorable moments spent at the university, particularly with Zeinab Bakhshi Valojerdi, Taufik Akbar Sitompul, Lan Van Dao, Nitin Desai and Václav Struhár. Together, we engaged in enriching discussions, delving into research ideas as well as candid conversations about our personal lives, and exploring diverse perspectives on histories, cultures, and traditions.

I am certainly indebted to my family for their support and understanding over the years. To Naveera, for her unwavering patience and support throughout this journey.

This work is supported by ABB robotics and has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 764785, FORA—Fog Computing for Robotics and the industrial postgraduate school Automation Region Research Academy (ARRAY++) funded by The Knowledge Foundation.

List of Publications

Papers included in this thesis[1]

Paper A: Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, and Thomas Nolte. *Multi-processor scheduling of elastic applications in compositional real-time systems.* In the Journal of Systems Architecture, Volume 122, 2022.

Paper B: Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, and Thomas Nolte. *Evaluating dispatching and scheduling strategies for firm real-time jobs in edge computing.* In the proceedings of the IEEE 49th Annual Conference of the IEEE Industrial Electronics Societ (IECON), 2023.

Paper C: Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, and Thomas Nolte. *Dispatching deadline constrained jobs in edge computing systems.* In the proceedings of the IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA), 2023.

Paper D: Shaik Mohammed Salman, Van-Lan Dao, Alessandro V. Papadopoulos, Saad Mubeen, and Thomas Nolte. *Scheduling firm real-time applications on the edge with single-bit execution time prediction.* In the proceedings of the IEEE 26th International Symposium on Real-Time Distributed Computing (ISORC), 2023.

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

Paper E: Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, and Thomas Nolte. *Taming tardiness on parallel machines: online scheduling with limited job information.* MRTC Report, MDU 2024.

Related publications, not included in this thesis

Shaik Mohammed Salman, Vaclav Struhar, Alessandro V. Papadopoulos, Moris Behnam, and Thomas Nolte. *Fogification of industrial robotic systems: research challenges.* In Proceedings of the Workshop on Fog Computing and the IoT, 2019.

Shaik Mohammed Salman, Taufik Sitompul, Alessandro V. Papadopoulos, Saad Mubeen, and Thomas Nolte. *Fog computing for augmented reality: trends, challenges and opportunities.* In the proceedings of the IEEE International Conference on Fog Computing (ICFC), 2020.

Shaik Mohammed Salman, Struhár, V., Bakhshi, Z., Dao, V. L., Desai, N., Papadopoulos, A. V., Nolte, T., Karagiannis, V., Schulte, S., Venito, A., & Fohler, G. *Enabling fog-based industrial robotics systems.* In the proceedings of the 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2020.

Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, and Thomas Nolte. *A systematic migration methodology for complex real-time software systems.* In the proceedings of the IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC), 2020.

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.* In the Journal of Systems Architecture, Volume 117, 2021.

Shaik Mohammed Salman, Saad Mubeen, Filip Marković, Alessandro V. Papadopoulos, and Thomas Nolte. *Scheduling elastic applications in composi-*

tional real-time systems. In the proceedings of 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2021.

Van-Lan Dao, and Shaik Mohammed Salman. *Deep neural network for indoor positioning based on channel impulse response.* In the proceedings of 27th International Conference on Emerging Technologies and Factory Automation (ETFA), 2022.

Contents

I

Thesis

Chapter 1

Introduction

Systems such as industrial robots and automated guided vehicles rely on algorithms whose functional correctness depends on the availability of computed results within certain time intervals [1]. Such systems are said to have real-time requirements and are referred to as real-time systems. Software that manages systems with real-time requirements has been historically deployed on a network of dedicated single-core hardware devices. More recent designs embrace an integrated multiprocessor deployment architecture. In both approaches, the hardware devices are part of the physical system (embedded) or are in proximity to the system they control and the environment in which the system operates.

On the other hand, the cloud computing [2] paradigm provides computing capabilities through large data centers which often are geographically distant from end users. Despite this distance, it has transformed the ability to scale and deploy software for a broad set of businesses through platform-as-a-service and infrastructure-as-a-service models. This has led to considerable interest among academics and the industry in investigating and replicating the benefits offered by cloud computing for use cases that are dependent on embedded architectures.

For instance, architectural paradigms such as fog computing have been developed that intend to provide resources in proximity to end users to provide low response times while acting as gateways to centralized cloud resources [3]. Realizing the benefits of such architectures requires addressing a myriad of

technical challenges [4, 5, 6]. These include developing protocols that ensure predictable communication latencies [7], virtualization techniques that ensure isolated access to hardware resources [8], orchestration approaches that improve utilization [9] and scheduling techniques that can satisfy computational latency requirements of real-time applications [10].

In this context, this thesis focuses on a subset of challenges related to scheduling techniques to satisfy certain real-time requirements, taking into account on-demand availability and elasticity of computational resources. Specifically, the thesis provides:

> ***Online*** *algorithms that satisfy* ***hard*** *real-time requirements of* ***elastic*** *applications in clairvoyant settings, improve* ***throughput*** *in non-clairvoyant settings for* ***firm*** *real-time applications and reduce average modified* ***tardiness*** *in non-clairvoyant settings for* ***soft*** *real-time applications.*

The design and evaluation of these algorithms is explored in detail in the papers included in this thesis. To guide the reader, the following section outlines the thesis structure.

Thesis outline: This thesis consists of two parts. Part I provides an overview of the thesis and is organized as follows. Chapter 2 provides background information on scheduling, while Chapter 3 presents an overview of the current state-of-art closely related to the work in this thesis. Chapter 4 discusses the research challenges addressed in this thesis. Chapter 5 summarizes the research methods used to develop the contributions presented in the thesis. Chapter 6 provides an outline of the included papers and their contributions. Lastly, Chapter 7 summarizes the thesis and introduces potential directions for future work. Part II of the thesis consists of the included papers.

Chapter 2

Background

As the work in this thesis addresses scheduling challenges, this chapter provides the relevant background with an overview of scheduling concepts (Section. 2.1) together with the considered multi-server configurations (Section. 2.2) and the variants of the online scheduling problem that depend on the knowledge of job processing times (Section. 2.3).

2.1 Scheduling Theory

Scheduling theory broadly deals with the problem of providing sufficient resources (*servers*) to requesters (*jobs*) such that certain performance criteria are satisfied. Addressing this problem involves controlling the order and the duration for which the jobs are allowed to access servers. Variants of this problem are investigated via tools such as mathematical modeling, analysis, and simulations. The parameters contributing to the different variants of this problem include:

- Number of servers and their types.
- Job characteristics such as their processing times and release times.
- The ability to preempt and migrate the jobs among the different servers.
- Knowledge of job characteristics at the time of decision making.

- Performance objectives such as minimization of completion times, flow times, tardiness, and throughput maximization.

Depending on the parameters and the objective, a decision-making entity, commonly referred to as a *scheduler*, assigns, at any time t, jobs to servers according to a scheduling policy that determines the job order and the server access duration. Scheduling policies may allow preemption and possibly migration, i.e., jobs can be paused and resumed later and possibly on a different server. Depending on the availability of the complete knowledge of the input job instance before processing begins, scheduling policies can be offline or online. A scheduling policy is optimal for a given problem instance if it provides the best possible solution while satisfying all the associated constraints. An approximate scheduling policy offers a feasible solution that may not be optimal but is within a bounded factor of an optimal solution.

Offline Scheduling: In the offline version of a scheduling problem, information about an input job instance is assumed to be available before executing the first job. This information includes knowledge of processing times and the arrival times of all jobs. The goal is to utilize this information to create an optimal or approximate schedule that establishes a mapping between jobs, servers, and the time intervals during which a server is made available to a specific job [11]. This problem is commonly addressed in application areas such as distributed embedded systems [1]. Solution techniques include constraint programming [12] and meta-heuristics like simulated annealing [13].

Online Scheduling: In the online version of a scheduling problem, jobs arrive over time. Schedulers assign jobs to servers and determine the processing order based on the information available at the time of decision making. They make no assumptions about future jobs and have no information about the future jobs [14]. Typical decision-making time points include job arrival times and job completion times with the possibility of additional decision time points depending on the algorithm and problem instance [15, 16]. Online scheduling finds applications in computer systems such as cloud computing and real-time embedded systems.

Competitive Ratio: The performance of an approximate solution is commonly measured as the ratio between the performance of an approximate solution and that of an optimal solution. For online problems, an often-used measure is the online competitive ratio.

It is defined as the worst-case ratio between the performance of the online algorithm and the optimal offline algorithm [17]. For a minimization problem, the online competitive ratio c is defined as:

$$c = \sup_{\sigma} \frac{C_{\text{online}}(\sigma)}{C_{\text{optimal}}(\sigma)}$$

where $C_{\text{online}}(\sigma)$ is the cost incurred by the online algorithm for input sequence σ, and $C_{\text{optimal}}(\sigma)$ is the cost incurred by the optimal offline algorithm for the same input sequence. An algorithm with an online competitive ratio of c is said to be c-competitive.

2.2 Multi-Server Configurations

The work in this thesis, in most cases, focuses on online scheduling in a parallel server environment in which all the servers are homogeneous; that is, a job takes the same amount of processing time to finish on any server. Since several multi-server configurations are considered in this thesis, an overview is provided below.

Centralized Single-Queue Multi-Server Systems: This configuration is a fundamental model in scheduling [18] involving the availability of multiple parallel servers. Here, incoming jobs either preempt an existing job or wait in a centralized single queue. At any given time, each server may process at most one job. A preempted job is returned to the queue and may resume its execution on possibly a different server. Some servers may remain idle if there are fewer jobs in the queue than there are servers.

Distributed Multi-Queue Multi-Server Systems: This configuration models computer systems in which each server has its own queue [18]. Here, incoming jobs are assigned to a server as soon as they arrive. A scheduling policy

at each server determines the order in which the assigned jobs waiting in their respective queues are processed. Additionally, preempted jobs are retained in the same queue as the server on which they were preempted and resume their execution later on the same server.

Hierarchical Multi-Server Systems: This configuration models systems that consist of at least one dedicated server and, at most, one fractional server. 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 [19]. The reserved computing time can be made available to the application through various reservation approaches such as the periodic server model [20] or the minimum-parallelism supply model [21]. Here, a local scheduler determines the order in which jobs of an application are ordered, while a global scheduler determines the scheduling order of the reserved servers. The job queuing and processing policies may follow a single-queue or distributed multi-queue approach.

On-Demand Multi-Server Systems: This configuration models computer systems in cloud-computing in which jobs are assigned to on-demand servers and a fixed set of servers. Here, on-demand servers refer to the computational resources that can be accessed by an application almost instantaneously and returned to a server pool after the application completes it processing. The usage of on-demand servers may vary in terms of the number of servers and the duration for which they are available. This model can be applied to all previously discussed configurations.

2.3 Online Scheduling Variants

As the work in this thesis focuses on online scheduling, this section briefly describes the different variants of the problem depending on the job processing time information available at the time of decision-making.

Clairvoyant Online Scheduling: In the clairvoyant version of the online problem, the processing time of a job is made available to the scheduler upon arrival. In this setting, the Shortest-Remaining-Processing-Time (SRPT) algorithm is optimal for minimizing mean response times on single machines[18]. Algorithms such as SRPT are not common in practice since obtaining precise processing time for individual jobs is generally difficult. Yet, this setting provides valuable insights into optimal online scheduling.

Semi-Clairvoyant Online Scheduling: In the semi-clairvoyant version of the online problem, the *class* of a job is made available to the scheduler upon arrival instead of its exact processing time. A job's *class* is defined as an integer i such that the actual processing time of the job is in the range $[2^i, 2^{i+1}]$. The Lowest-Class-First (LCF) algorithm is known to be optimal in this setting for the objective of minimizing mean response times [22].

Stochastic Online Scheduling: In the stochastic version of the online problem, it is assumed that the information about the distribution of a job's processing time is known to the scheduler on the arrival of the job, but the true processing time is known only at its completion time [11]. The *MinIncrease* algorithm is known to be optimal for minimizing mean response times in this setting [23].

Prediction Clairvoyant Online Scheduling: In the prediction clairvoyant version of the online problem, it is assumed that the predicted value (or a proxy) of a job's processing time is available to the scheduler on the arrival of a job. The scheduler can use this information to decide the order in which the jobs are processed. It has been shown that even single-bit classifiers that distinguish long jobs from short jobs can significantly improve performance for minimizing mean response times [24].

Non-clairvoyant Online Scheduling: In the non-clairvoyant version of the online problem, it is assumed that no information about the processing time is available for an incoming job. A scheduler has to decide the order in which

incoming jobs are processed without knowing how long they will take to execute. For single-server settings, the round-robin algorithm has been shown to be 4-competitive [25].

2.4 Performance Objectives

The performance objectives of the work in the thesis are related to real-time guarantees. In real-time theory, a widely considered performance objective is that of satisfying *hard* real-time guarantees. In this scenario, jobs must access servers such that each job completes its execution before its deadline. Even one job missing its deadline results in failure. Another performance objective relates to providing *soft* real-time guarantees. This metric relaxes the condition that all jobs must complete their execution within their deadlines to one in which some jobs can miss their deadlines. Additionally, all jobs must complete their execution even if they have missed their deadlines. A third type of performance objective relates to providing *firm* real-time guarantees. This objective not only allows some jobs to miss their deadlines but also requires jobs that missed their deadlines to be removed from the queue of jobs that need access to servers.

The contributions of this thesis address each of the objectives outlined above. To further facilitate a comprehensive understanding of these performance objectives, the following definitions are essential:

Definition 1. *Tardiness (T_j) is defined as the maximum of zero or the difference between the completion time and the deadline of a job j, i.e.,*

$$T_j = \max(C_j - d_j, 0)$$

Definition 2. *Modified Tardiness ($\tilde{T_j}$) is defined as the maximum between completion time and deadline of a job j. i.e.,*

$$\tilde{T_j} = \max(C_j, d_j)$$

Definition 3. *Throughput is defined is as the ratio of the number of jobs that finish within their deadlines and the total number of released jobs in a measurement interval.*

With these definitions in place, the performance objectives considered in this thesis can now be formally stated.

Zero Tardiness: For applications with hard real-time guarantees, the objective is to ensure that all jobs finish within their deadlines; that is, given an instance of jobs J, the number of tardy jobs should be equal to zero, i.e., $\sum(U_j) = 0$, where U_j is defined as:

$$U_j = \begin{cases} 1 & \text{if } C_j > d_j, \\ 0 & \text{otherwise} \end{cases}$$

where C_j is the completion time of job j and d_j is its deadline.

Average Throughput Maximization: For applications with firm real-time guarantees, the objective is maximizing the average number of jobs that finish within their deadlines, given job arrival and processing time distributions. This can be expressed as:

$$\mathbf{E}\left[\sum_{j \in J}(1 - U_j)\right]$$

Total Modified Tardiness Minimization: Another tardiness-related objective for applications with soft real-time guarantees is minimizing the total modified tardiness of jobs. This can be expressed as:

$$\min \sum_{j \in J} \tilde{T}_j$$

Chapter 3

Related Work

Since the work in this thesis focuses on challenges related to scheduling with real-time requirements on multi-server systems, this chapter summarizes the current state-of-art closely related to the contributions in this thesis. This includes work related to ensuring zero tardiness in hierarchical multi-server systems (Section. 3.1), maximizing throughput in multi-server systems (Section. 3.2), minimizing tardiness in multi-server systems (Section. 3.3) and results related to scheduling with predictions (Section 3.4).

3.1 Hierarchical Scheduling in Multi-Server Systems

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 [19]. The reserved computing time can be made available to the application through various reservation servers such as the periodic server and the deferrable server [20]. The mechanisms for defining such an interface and the schedulability tests vary depending on the scheduling strategies used for both local and global schedulers.

For single-server systems, Davis and Burns [20] provided an exact schedulability test for a hierarchical system with Fixed-Priority preemptive schedulers (FP) for local and global scheduling. They evaluated the schedulability under

periodic servers, sporadic servers, and deferrable servers. They concluded that the periodic servers provide better schedulability than the deferrable and sporadic servers. Similarly, Zhang and Burns [26] 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 [27] proposed the regularity-based resource supply model and provided schedulability conditions for applications where either FP or EDF was used as the local scheduler. Shin and Lee [28] proposed the periodic resource supply model to define the guaranteed resource time for an application and the schedulability tests when FP or EDF is used as the local scheduling policy. Easwaran et al. [29] generalized the periodic resource model and provided methods to generate the optimal bandwidth interfaces and improve resource utilization. Dewan and Fisher [30, 31] proposed an algorithm to determine the optimal server parameters for the periodic resource model while Kim et al. [32] proposed a method to reduce the overhead associated with the periodic resource model. Yang and Dong [33] considered scheduling mixed-criticality tasks within a periodic resource model where each resource supply reservation had multiple bandwidth estimates.

For multi-server reservations, Leontyev and Anderson [34] proposed the minimum-parallelism supply model to schedule applications with soft real-time requirements. Yang and Anderson [21] provided conditions to preserve the optimality of the minimum-parallelism supply model for hard real-time applications. Easwaran et al. [35] extended the periodic resource model with an additional parameter that specifies the maximum number of physical servers 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 scheduling policy [36]. In addition, they provided a transformation to generate equivalent periodic tasks for multi-server resource reservations (MPR) to be scheduled at the system level. Burmaykov et al. [37] proposed a generalization of the MPR model to reduce bandwidth allocation pessimism and provided schedulability conditions for global EDF and global FP scheduling policies. Pathan et al. [38] proposed an overhead-aware interface generation method for multi-server reservations with global FP scheduling policies.

Khalilzad et al. [39, 40, 41] 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. [42] considered a similar approach to variable task management in which bandwidth is adjusted so that each server 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. [43] provided an adaptive scheme similar to the work presented in this thesis for applications with a finite set of modes and also consider power consumption as a constraint.

3.2 Throughput Maximization in Multi-Server Systems

Centralized Single-Queue Multi-Server Systems: When the number of machines is fixed, the problem of online throughput maximization has been extensively studied, especially in the clairvoyant and stochastic settings for both single and multiple servers. A well-known result is the optimality of the EDF algorithm for a set of feasible jobs on a single machine. Here, optimality refers to the fact that given any feasible set of jobs schedulable on a single machine, EDF schedules the jobs without missing any deadlines. For centralized single-queue multi-server systems where all jobs have unit weights, Phillips et al. [44] showed that EDF and Least-Laxity-First (LLF) are $(2 - \frac{1}{m})$-speed algorithms where m is the number of machines. If the objective is zero tardiness and extra machines are allowed compared to an offline algorithm, they showed that EDF needs at most $O(P)$ extra machines, where P is the ratio of the maximum and minimum processing times of the jobs in the instance whereas LLF needs at most $O(\log P)$ extra machines. Chen et al. [45] improved this and provided an $O(\log m)$-optimal algorithm. Azar [46] provided an improved algorithm that requires $O(\frac{\log m}{\log \log m})$ extra machines. Im et al. [47] further improved this result and provided an algorithm that needs $O(\log \log m)$ extra machines.

In firm real-time settings (i.e., when some jobs can be abandoned), Kalyanasundaram and Pruhs [48] provided a randomized algorithm that is O(1)-competitive for the problem of throughput maximization. For instances where all jobs have some slackness ϵ, Lucier et al. [49] provide $O(1 + \epsilon)$-speed, $O(\frac{1}{\epsilon})$-competitive algorithm. Moseley et al. [50] remove the require-

ment of slackness and provide the best known O(1)-competitive algorithm for this problem.

For a more general version of the problem where each job has a weight, Canetti and Irani [51] showed a $\Omega\left(\sqrt{\frac{\log k}{\log \log k}}\right)$ lower bound for any randomized algorithm where k is the minimum between the maximum job value, the maximum job duration, and the ratio between the maximum and minimum job value-density. Azar [52] showed an improved lower bound of $\Omega(\log k)$. Eberle [53] provided a $(\frac{1}{\epsilon})$-competitive algorithm for the case where jobs have slackness.

For the stochastic version of the problem, Abhaya et al. [54] provided a method to calculate the mean delay for the M/G/1/ queuing systems. Kargahi et al. [55] provided bounds for deadline miss probabilities for the M/M/k systems. Bryant et al. [56] showed that an EDF-based policy which postpones execution of certain jobs was optimal for scheduling on related machines when the total system load was less than one and jobs have exponentially distributed processing times.

Distributed Multi-Queue Multi-Server Systems: For distributed multi-queue multi-server systems, Kargahi et al. [57] considered stochastic online scheduling settings where jobs at individual servers are ordered according to the EDF policy. These jobs are assumed to be exponentially distributed with firm real-time requirements. They provided an approach to estimate deadline miss probabilities. As case studies, they considered three routing policies: Join-Shortest-Queue (JSQ), Minimal Expected Unfinished Work (MED), and a Threshold-Based Policy (TBP). JSQ assigns incoming jobs to the server with the least number of pending jobs. MED is a generalization of JSQ for heterogeneous servers, which assigns incoming jobs to the server with minimal expected unfinished work. With TBP, the assignment probabilities depend on a threshold for the number of pending jobs at individual servers.

3.3 Tardiness Minimization in Multi-Server Systems

Centralized Single-Queue Multi-Server Systems: The problem of minimizing tardiness even in offline settings and on a single machine is known to be

NP-hard [58]. Due to inherent difficulty in developing competitive algorithms for this problem a modified tardiness objective of the form $\sum w_j(T_j + d_j)$ was considered. This modified objective was introduced in the offline version of the problem with unit weights in which all jobs have a common deadline by Kovalyov and Werner [59]. Kolliopoulos and Steiner [60] considered the general version of the problem and showed a reduction to the problem of finding an approximate solution to the problem of weighted total completion time. Their result showed that any ρ-approximation algorithm for the problem of minimizing total weighted completion time was an $(\rho+1)$-approximation algorithm for the problem of minimizing weighted modified total tardiness. Liu et al. [61] extended this idea to online settings in addition to an availability constraint. They provided $O(1)$-competitive algorithms for clairvoyant scenarios. Specifically, for the single-machine version of the problem with weights, they showed a 2-competitive lower bound and a 3-competitive algorithm as an upper bound. For the multiple-machine version of the unit weight problem, they provided a lower bound of 1.309 and a 3-competitive algorithm. For the distributed multi-queue multi-server version of the problem that takes into account on-demand servers, Nakahira et al. [62] provided algorithms that minimized the mean and variance of the service capacity while minimizing the tardiness costs. In each of these scenarios, it is assumed that the knowledge of the processing time of a job is available at its release time. For the stochastic version of the problem, where the processing times are exponentially distributed, Bryant et al. [56] showed that the earliest-deadline-first was optimal for scheduling on multiple machines when the total system load was less than one.

Distributed Multi-Queue Multi-Server Systems: In multi-server environments, dispatching policies determine the server on which an incoming job will be executed. In online dispatching, the dispatching decision is made when the job arrives at a dispatching server. Dispatching policies, such as JSQ, and variants, such as Join-Shortest-Queue among k randomly chosen servers (JSQ-k), need to know the number of pending jobs in each of the considered servers [18]. Gathering this information may take significant time, depending on the number of servers and the network traffic. As an alternative, policies such as round-robin are agnostic to pending jobs on the servers and dispatch incoming jobs to

the servers in a repeating pattern. The advantage of policies such as round-robin is that they do not have the overhead associated with policies that require information about the pending workload. In most existing work, the objective of dispatching policies has been to minimize mean flow time. Several additional policies, such as join-the-idle-queue and join-below-threshold, where servers notify the dispatchers when they are idle or have pending jobs less than a predefined value, have been proposed to balance the trade-off between overheads and flow time [63, 64].

Hyytiä and Righter [65] considered the problem of routing jobs with known processing times and deadlines in distributed multi-queue multi-server systems. They consider the case where all jobs have a single waiting time deadline, and each server schedules jobs based on their arrival order. Additionally, jobs that miss their deadlines still need to complete. They developed a deadline-aware dispatching policy that outperforms the Least-Work-Left (LWL) dispatching policy concerning deadline violation probability for low-variance job processing time distributions and high-variance distributions. The dispatching policy assigns jobs to servers for which the expected weighted tardiness is the least. Hyytiä et al. [66] consider a slightly different problem in which all jobs have the same processing time requirements, but each job has a unique waiting time deadline. They developed a dispatching policy that relies on the first policy iteration of the random dispatch policy and showed that this policy performs best to minimize expected weighted tardiness. They further considered a generalized version of this problem in which job processing times are discrete. That is, the job processing times are random multiples of some concrete processing time value [67]. Here, too, their policy outperforms JSQ and LWL dispatching policies. Wang et al. [68] provided a load balancing and a core allocation strategy to minimize mean flow time on heterogeneous servers with both reserved and on-demand servers. Here, the on-demand servers are utilized when the queue is full or the waiting time exceeds a predefined maximum waiting time threshold.

3.4 Scheduling with Predictions

Much of the discussed related work assumes clairvoyant settings in which the job processing times are known or assume knowledge of their distributions.

In several practical scenarios, such information is hard to come by. To address this, non-clairvoyant algorithms in deterministic and randomized settings have been developed to minimize completion time and flow time objectives. Motwani et al. [14] presented a $\Omega(P)$-competitive deterministic algorithm to minimize total flow time on multiple servers. Bechhetti and Leonardi [69] developed a randomized version of the multi-level feedback algorithm which is $O(min\{\log n, \log \frac{n}{m}, \log n \log P\})$-competitive (Given n jobs and m machines). For semi-clairvoyant settings, they developed a $O(\log P)$-competitive algorithm. Several works have investigated the possibility of improving the performance of algorithms with machine-learned advice or predictions, including classical algorithms targeting problems such as online scheduling and load-balancing [70]. The evaluation of these prediction-augmented algorithms has been done in terms of competitive analysis under accurate predictions and for possibly incorrect predictions [24, 64, 71, 72]. Some algorithms rely on predicting job processing times [64, 73] while others rely on predicting job orders [72].

Since predictions are most likely to be erroneous, Scully et al. [73] showed that in an $M/G/1$ system, the predicted shortest job first algorithm had an approximation ratio bounded by the parameters α, β when estimated processing time of a job j with actual processing time p_j was within an interval $[\alpha p_j, \beta p_j]$ and Akbari et al. [74] developed a simple dynamic priority scheme that relies on estimated job processing times and showed that their algorithm performed quite well for the same objective of minimizing mean flow time for low variance prediction errors. Mitzenmacher [24] studied the impact of single-bit predictors that can indicate if a job's processing time is above or below some threshold in the context of large-scale queuing systems for the performance metric of mean flow time and showed that such predictors can provide benefits similar to those achievable with the knowledge exact processing times for Poisson arrivals and certain processing time distributions. The analysis identified the impact of incorrect predictions and highlighted the improvements achieved even with such incorrect predictions against an algorithm that chooses a queue with the least number of pending jobs. Similarly, they extended their analysis for the case where the individual job processing times are predicted and showed via simulations that the benefits of the *supermarket model* in large distributed systems were retained if the predictions were *reasonably precise* [71]. Based on the

evaluations, they recommended choosing the queue with least number of pending jobs and using the preemptive shortest predicted job first policy for ordering jobs on a single server for use in actual systems, as this combination performed well in a diverse range of scenarios.

Zhao et al. [75] extended the Randomized Multi-Level Feedback algorithm (RMLF) to use predicted job processing times to minimize mean flow time. Their experimental evaluation shows that the prediction-based algorithm achieves performance close to SRPT's when the prediction error is small. If the error is large, their algorithm can achieve performance no worse than RMLF. Azar et al. [76] developed a deterministic, non-migratory $O(\mu \log P)$-competitive algorithm for multiple servers. Unlike the algorithm developed by Zhao et al. [75], their algorithm requires no knowledge of the distortion parameter μ, which is the product of the ratio of maximum overestimation and maximum underestimation of true processing times. The work in this thesis is inspired by the findings of these studies and extends these approaches to objectives involving deadlines.

Chapter 4

Research Challenges

Scheduling algorithms have been extensively used to address the problem of managing resource access and the associated performance objectives in various settings. Nonetheless, architectural paradigms such as edge and fog computing, and the demands of emerging applications have created new challenges that require improved techniques to address these demands. This chapter provides an overview of the research challenges addressed in this thesis. The challenge related to ensuring zero tardiness is discussed in Section 4.1. The challenges related to throughput maximization and tardiness minimization are discussed in Section 4.2 and Section 4.3, respectively.

4.1 RC1: Zero Tardiness and Dynamic Demands

There exist classes of real-time applications where ensuring zero tardiness is of paramount importance. Key factors that facilitate this include approaches that identify and specify the amount of processing time required, the time intervals during which the processing time should be available, and implementations that provide this guaranteed amount of processing time in required intervals. A common practice to model such resource demand for some applications is to use the sporadic task model that specifies the demand of a single task τ_j using a triplet of the form $\{C_j, T_j, D_j\}$, where each element represents the worst-case processing time, the minimum inter-arrival time between consecutive requests

and the time before which the requested processing time should have been allocated, respectively. An application Γ then consists of a set of n such tasks $\{\tau_1, \cdots, \tau_n\}$. Using this information, one can analyze the schedulability of the application for algorithms such as earliest-deadline-first using associated schedulability tests [77].

Dynamic Computational Demand: For some systems such as industrial and mobile robots, the computational demand is influenced by the environment in which they operate. In static environments, it may be possible to precisely measure the demand, while in dynamic environments, the measurement might only be approximate. This demand variability can be observed both in the processing times and in the periodicity of the tasks [78]. For example, processing time variability can be due to different conditional branches in the code triggered by external events, while period variability can result from different control laws. Mode-based analysis [79, 80, 81] is an alternative approach to ensure the schedulability of applications with varying demands. In this approach, each mode is be defined by a specific set of tasks. When tasks exist in multiple different modes, they are distinguished by different processing times and periods. Schedulability is then evaluated for each mode and the transition intervals. In cases where such an approach is not possible, e.g., difficulty in identifying and defining different modes due to dynamic environments, or when schedulability within a mode cannot be guaranteed due to insufficient computational resources, it may be necessary to introduce some form of overload management to maintain the schedulability of the application [82, 83].

Elastic Task Model: The elastic task model [82, 84] provides an approach for specifying dynamic demands by allowing task parameters to be defined as a range of values rather than a single value as in the sporadic task model. It takes into account the possibility that task parameters 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, overload management algorithms attempt to keep the application schedulable by modifying the task parameters to be as close as possible to their desired values based on the elastic coefficient of individual tasks.

RQ 1: What strategies can reduce the frequency of reservation bandwidth changes while ensuring zero tardiness for real-time applications with dynamic computational demands?

Reservation-Based Scheduling: Reservation-based scheduling approaches [85, 86] allocate processing time for each task according to its reservation parameters and schedule reservation servers rather than task instances (jobs). They allocate a fixed amount of computation time (called a budget) to a task during certain time intervals (called server periods) to ensure that each job of a task is completed by its deadline. Reservation servers allow each task to run for the duration of its budget within a server period. If a task has not completed its execution at the end of its budget, it is blocked until the next server period. If a task is released before the end of its minimum inter-arrival time, it will not be processed until its next server period starts. This approach ensures that tasks that violate their timing specifications do not affect the temporal behavior of other tasks in the system [83]. Static allocation of budget and the period, however, might be unable to meet the performance needs of applications with dynamic computational demands [87]. Applications with multiple modes and varying computational demands are managed by changing resource reservations and ensuring their schedulability during transition intervals [81, 41, 88]. 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 considering their quality-of-service requirements [42]. 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 for applications with distinct modes in [43, 89]. These solutions aim 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. This thesis extends this idea to support applications that do not have distinct modes but are rather specified according to the elastic task model to enable the execution of such applications in shared multi-server systems. Concretely, the work in this thesis addresses the research question RQ 1.

4.2 RC2: Throughput Maximization and Dynamic Capacity

Zero tardiness objective is the norm for control algorithms that manage critical components of a real-time system, such as the emergency stop functionality for industrial robots. Other components with dynamic demands may be more robust to occasional dropped jobs and perhaps even benefit if newer job instances based on fresh inputs are processed over older instances based on possibly stale inputs, thus allowing incomplete job instances to be dropped to process new jobs. Since ensuring zero tardiness often relies on resource allocation based on worst-case demands that may occur with low probability in some scenarios, it may result in underutilization of these resources. Instead of allocating resources based on worst-case demands, allocating resources based on average demands may provide improved resource utilization at a low or moderate loss in performance. Additionally, new architectural paradigms such as serverless computing and microservices are attractive due to their scaling capabilities thanks to the almost instantaneous on-demand availability of computational resources in large data centers and possibly in smaller geographically closer distributed data centers. A possible challenge is to consider how to effectively utilize these on-demand servers, given that accessing these servers dynamically may incur some costs. Therefore, the second challenge in the thesis deals with the objective of throughput maximization for firm real-time applications with on-demand availability of servers.

Focus on Clairvoyance: Substantial effort has been made to address the problem of throughput maximization for settings in which the processing times of jobs are precisely known or have known distributions. For instance, when the number of servers is fixed, randomized and deterministic algorithms exist that are $O(1)$-competitive [48, 50]. For the general model of weighted throughput on heterogeneous servers with slackness ϵ, there is a $O(\frac{1}{\epsilon})$- competitive algorithm [53]. These algorithms rely on precise knowledge of a job's processing times on its arrival. For the stochastic model where processing times are exponentially distributed, a migratory EDF-based algorithm has been shown to be optimal with respect to average throughput if the system load is less than one

on related servers [56].

Dynamic Service Capacity: Very few results consider the on-demand availability of service capacity for the throughput maximization problem. The only known result appears to be the generalized exact distributed scheduler [62] to minimize the variance of service capacity for jobs with deadlines. A closely related setting appears to be the dynamic bin-packing problem where the objective is to minimize the usage time of bins when jobs occupy the bins for a fixed duration of time and depart once they reach their deadlines [90, 91, 92]. This can be considered as a distributed multi-queue multi-server system where each server schedules jobs in their arrival order and processes them non-preemptively. Jobs are assigned to already open servers only if they can be processed on that server before their deadlines, otherwise they are dispatched to an on-demand server. Unfortunately, these approaches rely on knowledge of job processing times. Therefore, the work in this thesis addresses the research question RQ 2.

> **RQ 2:** Which approaches can improve average throughput for firm real-time applications with limited job processing time information in on-demand distributed multi-queue multi-server systems?

4.3 RC3: Tardiness and Non-Clairvoyance

Another challenge that is addressed in the thesis relates to scenarios where a large amount of data that acts as input to control algorithms needs to be processed. For instance, controlling a mobile robot using a vision-based motion planning algorithm deployed on the cloud may require transmitting images from multiple sources to localize the robot's position and plan an obstacle-free path for it to navigate. In this scenario, it may be beneficial from an end-to-end perspective to continue running the request to completion even if the job is past its deadline. The delay due to re-transmission and re-computation may not be worthwhile. Therefore, the third challenge addressed in this thesis relates to the problem of tardiness minimization in multi-server systems.

RQ 3: What strategies can be employed to minimize modified total tardiness for soft real-time applications with limited job processing time information in centralized single-queue multi-server systems?

Focus on Clairvoyance: Additionally, several variants of the tardiness minimization problem have been investigated as discussed in Section 3.3. However, no optimal algorithm is known even in clairvoyant settings. To overcome the difficulty in developing competitive algorithms for this objective [60], a modified tardiness objective has been considered. For this modified objective, the best-known results appear to be a 3-competitive algorithm by Liu et al. [61]. The dispatching policies by Hyytiä et al. [65, 66, 67] perform the best for the original tardiness objective in distributed multi-queue multi-server systems. When on-demand servers are allowed, Nakahari et al. [62] provide an efficient algorithm for the considered settings. These approaches, however, depend on precise knowledge of the processing times of jobs. While no known result exists for the non-clairvoyant scenario, following the reduction approach in [60, 61], it is straightforward to show a $O(P)$-competitive upper bound for the non-preemptive run-to-completion algorithm by Motwani et al. [14]. Therefore, the third challenge addresses the research question RQ 3.

Chapter 5

Research Methodology

The thesis results were developed following the hypothetico-deductive method [93]. This method involves making hypotheses based on existing theories and then deducing the logical consequences of these hypotheses. The research process consisted of the following four steps:

- **Problem Definition** - This step involves understanding the field of the topic through a comprehensive literature review, state-of-art and state-of-practice studies to identify gaps. Based on this, the scope of the problem is defined, setting the boundaries for the research. As an outcome of this approach, the three research challenges were identified.

- **Idea Development** - This step involves the iterative development of solutions to the defined problem. It starts with brainstorming and ideation, as well as proposing and discussing various potential solutions. These ideas are then refined and developed further, considering the problem's constraints and requirements. The solutions in this thesis addressing the identified research challenges were designed following this approach.

- **Implementation** - This step involves converting the idea and theory into an artifact. The proposed solution is implemented, often through a combination of coding, experimentation, and prototyping. This step is crucial for testing the solution's performance and identifying any practical

Research Methods	**Research Question**
State-of-Art Study	RQ 1, RQ 2, RQ 3
State-of-Practice Study	RQ 1, RQ 2, RQ 3
Simulation	RQ 1, RQ 2
Algorithm Design and Analysis	RQ 3

Table 5.1: Mapping of research methods utilized to address research questions in this thesis.

issues that might arise. The algorithms presented in this thesis were implemented and evaluated using C++ programs.

- **Evaluation** - This step involves evaluating the idea and its implementation. The solution is tested against predefined criteria or benchmarks to assess its effectiveness. The results of these tests are then analyzed and interpreted to draw conclusions about the solution's performance. This step involves comparing other existing solutions to determine the relative strengths and weaknesses of the proposed solution. While most solutions in this thesis were evaluated using simulations, a few were evaluated using competitive analysis.

Chapter 6

Thesis Contributions

Building on the research challenges and questions outlined in the previous chapter, this chapter presents the thesis contributions that address scheduling in multi-server systems and the performance requirements of real-time applications. Section. 6.1 summarizes the contribution addressing the research question related to the zero tardiness objective. Section. 6.2 summarizes the contribution addressing research question related to average throughput improvement. Section. 6.3 summarizes the contribution addressing the research question related to tardiness minimization.

Table. 6.1 establishes the relationship between the included papers and the research questions formulated in Chapter 4.

6.1 C1: Minimizing Bandwidth Changes in Hierarchical Multi-Server Systems

Topic: This contribution addresses the research question **RQ 1** and forms the main content of Paper A. It addresses the challenges of resource allocation and scheduling in hierarchical multi-server systems where applications may have computational needs that vary over time. This work bridges the gap between traditional real-time scheduling theory and the demands of adaptive computing environments.

Contribution	RQ 1	RQ 2	RQ 3
C1	Paper A	-	-
C2	-	Paper B,C and D	-
C3	-	-	Paper E

Table 6.1: Mapping of Research Contributions and Research Questions

Goal: This contribution focuses on minimizing reservation bandwidth changes while maintaining zero tardiness in hierarchical multi-server systems where several independent real-time applications can share the servers. This goal has been defined to ensure that other independent applications running on the same hardware are minimally impacted. To achieve this goal, a scheduling approach has been developed that minimizes reservation bandwidth changes for elastic tasks while satisfying period change requests. The approach combines per-server utilization modification, task re-partitioning, and reservation bandwidth adjustment.

Approach: This work considers a system model where elastic tasks must execute within reservation servers designed according to the multiprocessor minimum-parallelism resource supply form [21, 34]. This reservation approach offers applications a fixed number of fully dedicated servers and, at most, one partial server, available according to the periodic resource supply model. Each reservation has a local scheduler that manages application tasks. This scheduler adheres to the partitioned EDF scheduling policy, with reservation parameters initially set based on the desired utilization and period of the application's tasks. The primary goal is to minimize the frequency of reservation bandwidth changes once the initial parameters are established. To achieve this, when a task requests a period change, the utilization modification algorithm focuses solely on the tasks sharing the server with the requesting task. It attempts to generate new periods that satisfy the constraints of these affected tasks. If the algorithm fails to find a solution, a two-step approach is employed:

First, an attempt is made to re-partition the tasks among the servers, including the partial server, using a reasonable allocation scheme [94]. If re-partitioning is unsuccessful, the reservation bandwidth is then considered for

modification. This modification may involve adjusting the partial server bandwidth or, if necessary, allocating a new fully dedicated server.

The effectiveness of this approach is demonstrated through an evaluation presented in paper A. The per-server utilization modification scheme can satisfy up to 94 percent of requests. When combined with the re-partitioning step, a 100 percent success rate for period change requests is achieved, highlighting the approach's robustness in managing dynamic task requirements while considering the impact on other co-located applications.

Potential Impact: This contribution has implications for practical applications and academic research. In practice, this approach can help ensure schedulability by adjusting the application demands when additional capacity is unavailable. From an academic perspective, this study provides a basis for further research on elastic task scheduling in compositional systems. It offers a reference point for comparing future adaptive scheduling algorithms.

Limitations: While this contribution remains valuable, some aspects could be addressed in future work. An area for potential improvement could be examining overhead costs associated with frequent task re-partitioning, which the current work does not explicitly address. Additionally, the approach currently focuses on partitioned EDF as the local scheduling policy, which may limit its applicability in systems using different scheduling algorithms. Another area of improvement could be the run-time complexity of the approach. While the run-time complexity of this approach's task period assignment algorithm is quadratic in the number of tasks, this design provides only a probabilistic and slight improvement by reducing the number of tasks that may need to be modified. A more efficient approach may be desirable.

6.2 C2: Improving Average Throughput in Distributed Multi-Server Systems

Topic: This contribution addresses the research question **RQ 2** and is provided in papers B, C, and D. The broader area of contribution of these papers is

real-time scheduling and load balancing in on-demand multi-server systems. It focuses on managing jobs with stochastic execution times, inter-arrival times, and deadlines in multi-server systems, emphasizing meeting firm real-time requirements without precise processing time information. This work bridges the gap between traditional real-time systems and emerging edge computing paradigms, addressing the challenges due to on-demand resource availability.

Goal: The goal of this research is to develop online algorithms to maximize average throughput for a distributed multi-queue multi-server system with on-demand resource availability when inputs arrive following a Poisson process; the jobs processing times are drawn from an exponential distribution. The studies compare prediction-based admission and scheduling policies against traditional approaches like EDF, assessing their performance in meeting deadlines. A key objective has been to investigate the effectiveness of on-demand resource allocation for satisfying firm real-time requirements, particularly in scenarios where job processing times are not fully known in advance.

Approach: In paper B, the approach involves considering a dual-bit job size predictor that classifies an incoming job into one of four job size classes: small, medium, large, and very large. This information is then used with a Preemptive Shortest-Job-Class-First (PSJF) scheduling policy. The policy orders jobs according to their job size classes, and jobs in each class are in the First-In-First-Out (FIFO) order. The evaluation indicates that FIFO and EDF policies outperform the prediction-based policy in under-loaded conditions. However, in overloaded scenarios, the prediction-based policy offers better performance.

In paper C, the framework integrates an on-arrival dispatcher with EDF scheduling for an on-demand distributed multi-queue multi-server system. The dispatching policy uses a schedulability test that estimates job response times based on the number of pending jobs in each queue and dispatches jobs to servers likely to meet their deadlines. The dispatcher employs a Join-Shortest-Queue (JSQ) policy for assigning jobs across the servers. Simulations demonstrate that this approach significantly enhances throughput. It outperforms the first-fit heuristic and systems using only reserved servers, especially when jobs are dispatched to on-demand servers.

In paper D, a single-bit execution time prediction-based admission policy for online dispatching and scheduling jobs in distributed multi-queue multi-server systems is explored. The performance of this policy is compared against mean-approximation and clairvoyant policies, using simulations to evaluate throughput under various loads. The results indicate that the single-bit prediction policy outperforms the mean approximation policy but remains less effective than the clairvoyant policy which has precise processing time information of each of the jobs.

Potential Impact: This contribution has the potential for impact in both practical and academic settings. In practice, the results could lead to improved resource management in edge computing platforms for real-time applications, enhancing performance for industrial embedded systems that require offloading to edge resources. Academically, this work advances the understanding of real-time scheduling in distributed edge computing environments, providing new insights into the trade-offs between prediction-based and traditional scheduling policies for throughput maximization.

Limitations: The contributions rely primarily on simulation-based evaluation, which may not capture all real-world complexities. Additionally, the impact of network delays and communication overhead is not fully explored, potentially limiting the applicability of the findings in highly distributed environments.

6.3 C3: Minimizing Total Tardiness in Centralized Multi-Server Systems

Topic: This contribution addresses the research question **RQ 3** for soft real-time applications in centralized single-queue multi-server systems and is the main content of Paper E. The research explores using predictions to minimize total tardiness, addressing scheduling challenges in environments where job characteristics may not be fully known in advance. This work bridges the gap

between traditional real-time scheduling and prediction based approaches, aiming to improve system performance as measured by job tardiness.

Goal: The primary goal of this research has been to develop and evaluate prediction-based scheduling algorithms that can effectively minimize total tardiness in soft real-time applications. A key objective has been to investigate how incorporating predictions can enhance the outcome of scheduling decisions in centralized multi-server environments, particularly when dealing with uncertainties in job processing times and arrival patterns.

Approach: Three algorithms are proposed in paper E. All three algorithms were originally designed for minimizing flow times and completion times which are different from the tardiness minimization objective considered in this thesis. These algorithms prioritize jobs based on different criteria, such as job classes based on predicted processing time, actual class of the job, and job density calculated considering weight, predicted speeds, and known processing times. The first algorithm is $O(\mu \log P)$-competitive for scheduling unit weight jobs on parallel identical machines in prediction-clairvoyant settings. Similarly, the second algorithm is $O(\log P)$-competitive when scheduling unit weight jobs on parallel identical machines in semi-clairvoyant settings while the third algorithm is $O(\mu)$-competitive for scheduling arbitrary weight jobs on parallel unrelated machines and clairvoyant processing times in speed-prediction settings.

Potential Impact: Academically, this work advances the understanding of prediction-based scheduling in real-time systems, providing new insights into the trade-offs between prediction accuracy and scheduling performance for soft real-time jobs. It establishes a baseline for further research on integrating machine learning techniques with real-time scheduling.

Limitations: The focus on centralized single-queue multi-server systems limits the applicability of the findings to distributed or decentralized architectures. Moreover, this contribution does not consider the computational over-

head of making and utilizing predictions in real-time scenarios, which could impact overall system performance.

6.4 Personal Contributions

My personal contributions in the papers included in this thesis are summarized based on CRediT[1] (Contributor Roles Taxonomy) definitions in Table 6.2.

CRediT Contribution	Paper A	Paper B	Paper C	Paper D	Paper E
Conceptualization	✓	✓	✓	✓	✓
Data curation	✓	✓	✓	✓	✓
Formal analysis	✓				✓
Funding acquisition					
Investigation	✓	✓	✓	✓	✓
Methodology	✓	✓		✓	✓
Project administration					
Resources					
Software	✓	✓	✓	✓	✓
Supervision					
Validation	✓	✓	✓	✓	✓
Visualization	✓	✓	✓	✓	✓
Writing – original draft	✓	✓	✓	✓	✓
Writing – review and editing	✓	✓	✓	✓	✓

Table 6.2: Personal contribution following CRediT in included papers

6.5 Included Papers

Paper A

Title: Multi-processor scheduling of elastic applications in compositional real-time systems [95]
Authors: Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, Thomas Nolte

[1] `https://credit.niso.org/`

Status: Published in Journal of Systems Architecture, (December 2021)
Abstract: Scheduling of real-time applications modeled according to the periodic and the sporadic task model under hierarchical and compositional real-time systems has been widely studied to provide temporal isolation among independent applications running on shared resources. However, for some real-time applications that are amenable to variation in their timing behavior, usage of these task models can result in pessimistic solutions. The elastic task model addresses this pessimism by allowing the timing requirements of an application's tasks to be specified as a range of values instead of a single value. Although the scheduling of elastic applications on dedicated resources has received considerable attention, there is limited work on scheduling such applications in hierarchical and compositional settings.

In this paper, we evaluate different earliest deadline first scheduling algorithms to schedule elastic applications in a minimum parallelism supply form reservation on a multiprocessor system. Our evaluation indicates that the proposed approach provides performance comparable to the current state-of-the-art algorithms for scheduling elastic applications on dedicated processors in terms of schedulability.

Paper B

Title: Evaluating Dispatching and Scheduling Strategies for Firm Real-Time Jobs in Edge Computing [96]
Authors: Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, Thomas Nolte
Status: Published in the proceedings of 49th Annual Conference of the IEEE Industrial Electronics Society (IECON 2023)
Abstract: We consider the problem of on-arrival dispatching and scheduling jobs with stochastic execution times, inter-arrival times, and deadlines in multi-server fog and edge computing platforms. In terms of mean response times, it has been shown that size-based scheduling policies, when combined with dispatching policies such as join-shortest-queue, provide better performance over policies such as first-in-first-out. Since job sizes may not always be known apriori, prediction-based policies have performed reasonably well. However, little is known about the performance of prediction-based policies for jobs with firm

deadlines. In this paper, we address this issue by considering the number of jobs that complete within their deadlines as a performance metric and investigate, using simulations, the performance of a prediction-based shortest-job-first scheduling policy for the considered metric and compare it against scheduling policies that prioritize based on deadlines (EDF) and arrival times (FIFO). The evaluation indicates that in under-loaded conditions, the prediction-based policy is outperformed by both FIFO and EDF policies. However, in overloaded scenarios, the prediction-based policy offers slightly better performance.

Paper C

Title: Dispatching deadline-constrained jobs in edge computing systems [97]
Authors: Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, Thomas Nolte
Status: Published in the proceedings of IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA 2023)
Abstract: The edge computing paradigm extends the architectural space of real-time systems by bringing the capabilities of the cloud to the edge. Unlike cloud-native systems designed for mean response times, real-time industrial embedded systems are designed to control a single physical system, such as a manipulator arm or a mobile robot, that requires temporal predictability. We consider the problem of dispatching and scheduling of jobs with deadlines that can be offloaded to the edge and propose DAL, a deadline-aware load balancing and scheduling framework that leverages the availability of on-demand computing resources along with an on-arrival dispatching scheme to manage temporal requirements of such offloaded applications. The evaluation indicates that DAL can achieve reasonably good performance even when execution times, arrival times, and deadlines vary.

Paper D

Title: Scheduling firm real-time applications on the edge with single-bit execution time prediction [98]
Authors: Shaik Mohammed Salman, Van Lan Dao, Alessandro V. Papadopoulos, Saad Mubeen, Thomas Nolte

Status: Published in the proceedings of IEEE 26th International Symposium on Real-Time Distributed Computing (ISORC 2023)
Abstract: The edge computing paradigm brings the capabilities of the cloud, such as on-demand resource availability, to the edge for applications with low latency and real-time requirements. While cloud-native load balancing and scheduling algorithms strive to improve performance metrics like mean response times, real-time systems that govern physical systems must satisfy deadline requirements. This paper explores the potential of an edge-computing architecture that utilizes the on-demand availability of computational resources to satisfy firm real-time requirements for applications with stochastic execution and inter-arrival times. As it might be difficult to know the precise execution times of individual jobs before completion, we consider an admission policy that relies on single-bit execution time predictions for dispatching. We evaluate its performance in terms of the number of jobs that complete by their deadlines via simulations. The results indicate that the prediction-based admission policy can achieve reasonable performance for the considered settings.

Paper E

Title: Taming Tardiness on Parallel Machines: Online Scheduling with Limited Job Information [99]
Authors: Shaik Mohammed Salman, Alessandro V. Papadopoulos, Saad Mubeen, Thomas Nolte
Status: MRTC Report. MDU 2024
Abstract: We consider the problem of scheduling n jobs with soft deadlines on M parallel machines in online settings. Since no bounded competitive algorithms exist to minimize total tardiness $\sum w_j T_j$, we consider an objective of the form $\sum w_j(T_j + d_j)$. For this modified objective, we provide competitive algorithms with different levels of information about job processing times.

Chapter 7

Conclusion

7.1 Conclusion

In this thesis, challenges in scheduling real-time applications in multi-server systems were addressed through the investigation of three key research questions. The first question investigated how the frequency of reservation bandwidth changes could be reduced while ensuring zero tardiness for real-time applications with dynamic computational demands, assuming a setting with complete knowledge of job processing times. The second question explored approaches to improve average throughput for firm real-time applications with limited job information in on-demand distributed multi-queue multi-server systems, given knowledge of job size distributions. Finally, the third question examined methods to minimize tardiness for soft real-time applications in centralized single-queue multi-server systems, specifically in non-clairvoyant settings where exact job processing times are unknown. For each of these research questions, solutions are proposed in the papers included in this thesis.

To address the first research question, a framework was developed to minimize reservation bandwidth changes in compositional multi-server systems while ensuring zero tardiness. For the second question, a framework was presented that aimed to improve average throughput for firm real-time applications in distributed multi-queue multi-server systems by utilizing processing time predictions. Finally, to address the third research question, a set of online

scheduling algorithms was introduced for soft real-time applications aimed at minimizing tardiness in centralized single-queue multi-server systems. These algorithms, characterized by their competitive ratios, utilized varying levels of information regarding processing times to achieve their objective.

7.2 Future Directions

Building on the insights derived from investigation of the research questions, some potential directions for future research can be identified. These directions could extend the findings presented and address further challenges in scheduling real-time applications within multi-server systems.

Zero Tardiness: Zero tardiness is a critical requirement in many real-time applications, particularly those that demand strict adherence to deadlines. One promising direction is to examine the effectiveness of the approach proposed in this thesis by employing local scheduling policies other than partitioned EDF. This investigation may provide valuable insights into how various scheduling algorithms interact with the elastic task model and reservation server framework. Additionally, considering an elastic parallel task model in a hierarchical multi-server system could represent a meaningful extension of the current work, contributing to a deeper understanding of task allocation and resource utilization in more complex environments.

Throughput Maximization: Maximizing throughput is essential for ensuring that systems can efficiently handle the workload of firm real-time applications, particularly in environments where resources are limited or variable. Another avenue worth exploring is the integration of machine learning techniques for enhancing the accuracy of execution time predictions, which could potentially improve the effectiveness of the proposed scheduling approaches. Developing theoretical bounds and analytical models for these strategies may offer a stronger foundation for understanding their behavior and limitations. Furthermore, extending the scope of this research to consider end-to-end latency in complex edge-cloud ecosystems could address broader system-level

integration challenges. Investigating the use of heterogeneous edge computing resources with varying processing capabilities may also yield insights into optimizing performance in more realistic hardware configurations.

Tardiness Minimization: Tardiness minimization is particularly important in soft real-time applications, where occasional deadline misses may be acceptable but should be kept to a minimum to maintain quality of service. The algorithms proposed in this thesis for minimizing tardiness primarily focus on the availability of a fixed number of servers. A potential direction for future work could involve integrating these scheduling algorithms with the concept of on-demand server availability, thereby offering more flexibility and responsiveness in dynamic environments. Additionally, exploring performance within a distributed multi-queue multi-server system could lead to scalable solutions and improved efficiency in resource allocation. Through the investigation of these areas, future research could build upon the findings of this thesis and contribute to the development of effective scheduling strategies for real-time applications.

Bibliography

[1] Hermann Kopetz. *The Real-Time Environment*, pages 1–28. Springer US, Boston, MA, 2011.

[2] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. *National institute of science and technology, special publication*, 800(2011):145, 2011.

[3] Michaela Iorga, Larry Feldman, Robert Barton, Michael J Martin, Nedim S Goren, Charif Mahmoudi, et al. Fog computing conceptual model. Technical report, National Institute of Standards and Technology, 2018.

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

[5] Shaik Mohammed Salman, Taufik Akbar Sitompul, Alessandro Vittorio Papadopoulos, and Thomas Nolte. Fog computing for augmented reality: Trends, challenges and opportunities. In *2020 IEEE International Conference on Fog Computing (ICFC)*, pages 56–63. IEEE, 2020.

[6] Mohammed Salman Shaik, Václav Struhár, Zeinab Bakhshi, Van-Lan Dao, Nitin Desai, Alessandro V Papadopoulos, Thomas Nolte, Vasileios Karagiannis, Stefan Schulte, Alexandre Venito, et al. Enabling fog-based industrial robotics systems. In *2020 25th IEEE International Conference*

on Emerging Technologies and Factory Automation (ETFA), volume 1, pages 61–68. IEEE, 2020.

[7] Victor Millnert, Johan Eker, and Enrico Bini. Achieving predictable and low end-to-end latency for a network of smart services. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–7. IEEE, 2018.

[8] Michael Pearce, Sherali Zeadally, and Ray Hunt. Virtualization: Issues, security threats, and solutions. *ACM Computing Surveys (CSUR)*, 45(2):1–39, 2013.

[9] Václav Struhár, Silviu S Craciunas, Mohammad Ashjaei, Moris Behnam, and Alessandro V Papadopoulos. Hierarchical resource orchestration framework for real-time containers. *ACM Transactions on Embedded Computing Systems*, 23(1):1–24, 2024.

[10] Mor Harchol-Balter. Open problems in queueing theory inspired by datacenter computing. *Queueing Systems*, 97(1):3–37, 2021.

[11] Michael L. Pinedo. Offline deterministic scheduling, stochastic scheduling, and online deterministic scheduling. In Joseph Y.-T. Leung, editor, *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.

[12] Mohammadreza Barzegaran and Paul Pop. Communication scheduling for control performance in tsn-based fog computing platforms. *IEEE Access*, 9:50782–50797, 2021.

[13] Paul Pop, Konstantinos Alexandris, and Tongtong Wang. Configuration of multi-shaper time-sensitive networking for industrial applications. *IET Networks*, 2024.

[14] Rajeev Motwani, Steven Phillips, and Eric Torng. Nonclairvoyant scheduling. *Theoretical computer science*, 130(1):17–47, 1994.

[15] Susanne Albers. Online scheduling. In *Introduction to scheduling*, pages 71–98. CRC Press, 2009.

[16] Kirk Pruhs, Jirí Sgall, and Eric Torng. Online scheduling. In Joseph Y.-T. Leung, editor, *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.

[17] Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. cambridge university press, 2005.

[18] Mor Harchol-Balter. *Performance modeling and design of computer systems: queueing theory in action*. Cambridge University Press, 2013.

[19] 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.

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

[21] Kecheng Yang and James H. Anderson. On the Dominance of Minimum-Parallelism Multiprocessor Supply. *Proceedings - Real-Time Systems Symposium*, 0:215–226, 2016.

[22] Luca Becchetti, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Kirk Pruhs. Semi-clairvoyant scheduling. *Theoretical computer science*, 324(2-3):325–335, 2004.

[23] Nicole Megow, Marc Uetz, and Tjark Vredeveld. Stochastic online scheduling on parallel machines. In *Approximation and Online Algorithms: Second International Workshop, WAOA 2004, Bergen, Norway, September 14-16, 2004, Revised Selected Papers 2*, pages 167–180. Springer, 2005.

[24] Michael Mitzenmacher. Queues with small advice. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA21)*, pages 1–12. SIAM, 2021.

[25] Benjamin Moseley and Shai Vardi. The efficiency-fairness balance of round robin scheduling. *Operations Research Letters*, 50(1):20–27, 2022.

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

[27] 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.

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

[29] Arvind Easwaran, Insup Lee, Insik Shin, and Oleg Sokolsky. Compositional schedulability analysis of hierarchical real-time systems. In *ISORC*, pages 274–281. IEEE Computer Society, 2007.

[30] 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.

[31] 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.

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

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

[34] Hennadiy Leontyev and James H. Anderson. A hierarchical multiprocessor bandwidth reservation scheme with timing guarantees. *Proceedings - Euromicro Conference on Real-Time Systems*, pages 191–200, 2008.

[35] Arvind Easwaran, Insup Lee, Oleg Sokolsky, and Steve Vestal. A compositional scheduling framework for digital avionics systems. In *RTCSA*, pages 371–380. IEEE Computer Society, 2009.

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

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

[38] 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.

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

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

[41] Nima Khalilzad, Fanxin Kong, Xue Liu, Moris Behnam, and Thomas Nolte. A feedback scheduling framework for component-based soft real-time systems. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 182–193, 2015.

[42] Stefan Groesbrink, Luis Almeida, Mario De Sousa, and Stefan M. Petters. Towards certifiable adaptive reservations for hypervisor-based virtualization. *Real-Time Technology and Applications - Proceedings*, 2014-October(October):13–24, 2014.

[43] Tommaso Cucinotta, Luigi Palopoli, Luca Abeni, 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.

[44] Cynthia A Phillips, Cliff Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 140–149, 1997.

[45] Lin Chen, Nicole Megow, and Kevin Schewior. An o(m)-competitive algorithm for online machine minimization. *SIAM Journal on Computing*, 47(6):2057–2077, 2018.

[46] Yossi Azar and Sarel Cohen. An improved algorithm for online machine minimization. *Operations Research Letters*, 46(1):128–133, 2018.

[47] Sungjin Im, Benjamin Moseley, Kirk Pruhs, and Clifford Stein. An o (log log m)-competitive algorithm for online machine minimization. In *2017 IEEE Real-Time Systems Symposium (RTSS)*, pages 343–350. IEEE, 2017.

[48] Bala Kalyanasundaram and Kirk R Pruhs. Maximizing job completions online. *Journal of Algorithms*, 49(1):63–85, 2003.

[49] Brendan Lucier, Ishai Menache, Joseph Naor, and Jonathan Yaniv. Efficient online scheduling for deadline-sensitive jobs. In *Proceedings of the twenty-fifth annual ACM symposium on Parallelism in algorithms and architectures*, pages 305–314, 2013.

[50] Benjamin Moseley, Kirk Pruhs, Clifford Stein, and Rudy Zhou. A competitive algorithm for throughput maximization on identical machines. *Mathematical Programming*, pages 1–18, 2024.

[51] Ran Canetti and Sandy Irani. Bounding the power of preemption in randomized scheduling. In *Proceedings of the twenty-seventh annual ACM symposium on Theory of computing*, pages 606–615, 1995.

[52] Yossi Azar and Oren Gilon. Scheduling with deadlines and buffer management with processing requirements. *Algorithmica*, 78:1246–1262, 2017.

[53] Franziska Eberle. O $(1/\varepsilon)$ is the answer in online weighted throughput maximization. In *41st International Symposium on Theoretical Aspects of Computer Science (STACS 2024)*. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[54] Vidura Gamini Abhaya, Zahir Tari, Panlop Zeephongsekul, and Albert Y. Zomaya. Performance analysis of edf scheduling in a multi-priority preemptive M/G/1 queue. *IEEE Transactions on Parallel and Distributed Systems*, 25(8):2149–2158, 2014.

[55] Mehdi Kargahi and Ali Movaghar. A method for performance analysis of earliest-deadline-first scheduling policy. *Journal of Supercomputing*, 37(2):197–222, 2006.

[56] Richard Bryant, Peter Lakner, and Michael Pinedo. On the optimality of the earliest due date rule in stochastic scheduling and in queueing. *European Journal of Operational Research*, 298:202–212, 4 2022.

[57] Mehdi Kargahi and Ali Movaghar. Dynamic routing of real-time jobs among parallel edf queues: A performance study. *Computers & Electrical Engineering*, 36(5):835–849, 2010.

[58] Richard K Congram, Chris N Potts, and Steef L van de Velde. An iterated dynasearch algorithm for the single-machine total weighted tardiness scheduling problem. *INFORMS Journal on Computing*, 14(1):52–67, 2002.

[59] Mikhail Y Kovalyov and Frank Werner. Approximation schemes for scheduling jobs with common due date on parallel machines to minimize total tardiness. *Journal of Heuristics*, 8:415–428, 2002.

[60] Stavros G Kolliopoulos and George Steiner. Approximation algorithms for scheduling problems with a modified total weighted tardiness objective. *Operations research letters*, 35(5):685–692, 2007.

[61] Ming Liu, Yinfeng Xu, Chengbin Chu, and Feifeng Zheng. Online scheduling to minimize modified total tardiness with an availability constraint. *Theoretical computer science*, 410(47-49):5039–5046, 2009.

[62] Yorie Nakahira, Andres Ferragut, and Adam Wierman. Generalized exact scheduling: A minimal-variance distributed deadline scheduler. *Operations Research*, 71(2):433–470, 2023.

[63] Xingyu Zhou, Fei Wu, Jian Tan, Yin Sun, and Ness Shroff. Designing low-complexity heavy-traffic delay-optimal load balancing schemes: Theory to algorithms. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 1(2):1–30, 2017.

[64] Yecheng Zhao, Runzhi Zhou, and Haibo Zeng. Design optimization for real-time systems with sustainable schedulability analysis. *Real-Time Systems*, 58(3):275–312, 2022.

[65] Esa Hyytiä and Rhonda Righter. Routing jobs with deadlines to heterogeneous parallel servers. *Operations Research Letters*, 44(4):507–513, 2016.

[66] Esa Hyytiä, Rhonda Righter, Olivier Bilenne, and Xiaohu Wu. Dispatching fixed-sized jobs with multiple deadlines to parallel heterogeneous servers. *Performance Evaluation*, 114:32–44, 2017.

[67] Esa Hyytiä, Rhonda Righter, Olivier Bilenne, and Xiaohu Wu. Dispatching discrete-size jobs with multiple deadlines to parallel heterogeneous servers. *Systems modeling: methodologies and tools*, pages 29–46, 2019.

[68] Shuang Wang, Xiaoping Li, Quan Z Sheng, Ruben Ruiz, Jinquan Zhang, and Amin Beheshti. Multi-queue request scheduling for profit maximization in iaas clouds. *IEEE Transactions on Parallel and Distributed Systems*, 32(11):2838–2851, 2021.

[69] Luca Becchetti and Stefano Leonardi. Non-clairvoyant scheduling to minimize the average flow time on single and parallel machines. In *Proceedings of the thirty-third annual ACM symposium on Theory of computing*, pages 94–103, 2001.

[70] Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ml predictions. *Advances in Neural Information Processing Systems*, 31, 2018.

[71] Michael Mitzenmacher and Matteo Dell'Amico. The supermarket model with known and predicted service times. *IEEE Transactions on Parallel and Distributed Systems*, 33:2740–2751, 11 2022.

[72] Alexander Lindermayr and Nicole Megow. Permutation predictions for non-clairvoyant scheduling. In *Proceedings of the 34th ACM Symposium on Parallelism in Algorithms and Architectures*, SPAA '22, page 357–368, New York, NY, USA, 2022. Association for Computing Machinery.

[73] Ziv Scully, Isaac Grosof, and Michael Mitzenmacher. Uniform bounds for scheduling with job size estimates. *arXiv preprint arXiv:2110.00633*, 2021.

[74] Maryam Akbari-Moghaddam and Douglas G Down. Seh: Size estimate hedging scheduling of queues. *ACM Transactions on Modeling and Computer Simulation*, 2023.

[75] Tianming Zhao, Wei Li, and Albert Y. Zomaya. Real-time scheduling with predictions. In *2022 IEEE Real-Time Systems Symposium (RTSS)*, pages 331–343, 2022.

[76] Yossi Azar, Eldad Peretz, and Noam Touitou. Distortion-Oblivious Algorithms for Scheduling on Multiple Machines. In *Leibniz International Proceedings in Informatics, LIPIcs*, volume 248. Schloss Dagstuhl-Leibniz-Zentrum fur Informatik GmbH, Dagstuhl Publishing, 12 2022.

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

[78] 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.

[79] 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.

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

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

[82] Giorgio C. Buttazzo, Giuseppe Lipari, and Luca Abeni. Elastic task model for adaptive rate control. In *Proceedings - Real-Time Systems Symposium*, pages 286–295. IEEE, 1998.

[83] Giorgio Buttazzo, Giuseppe Lipari, Luca Abeni, and Marco Caccamo. *Soft Real-Time Systems*. Springer, 2005.

[84] 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.

[85] 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.

[86] Rafia Inam. *Hierarchical scheduling for predictable execution of real-time software components and legacy systems*. PhD thesis, Mälardalen University, December 2014.

[87] 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.

[88] Wei-Ju Chen, Peng Wu, Pei-Chi Huang, Aloysius K Mok, and Song Han. Online reconfiguration of regularity-based resource partitions in cyber-physical systems. *Real-Time Systems*, pages 1–44, 2021.

[89] 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.

[90] Yusen Li, Xueyan Tang, and Wentong Cai. On dynamic bin packing for resource allocation in the cloud. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*, pages 2–11, 2014.

[91] Yusen Li, Xueyan Tang, and Wentong Cai. Dynamic bin packing for on-demand cloud resource allocation. *IEEE Transactions on Parallel and Distributed Systems*, 27(1):157–170, 2015.

[92] Yossi Azar and Danny Vainstein. Tight bounds for clairvoyant dynamic bin packing. *ACM Transactions on Parallel Computing (TOPC)*, 6(3):1–21, 2019.

[93] Gordana Dodig-crnkovic. Scientific methods in computer science. In *In Proc. PROMOTE IT 2002, 2nd Conference for the Promotion of Research in IT at New Universities and at University Colleges in*, pages 22–24, 2002.

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

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

[96] Shaik Mohammed Salman, Alessandro Vittorio Papadopoulos, Saad Mubeen, and Thomas Nolte. Evaluating dispatching and scheduling strategies for firm real-time jobs in edge computing. In *IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society*, pages 1–6. IEEE, 2023.

[97] Shaik Mohammed Salman, Alessandro Vittorio Papadopoulos, Saad Mubeen, and Thomas Nolte. Dispatching deadline constrained jobs in

edge computing systems. In *2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 1–8. IEEE, 2023.

[98] Shaik Mohammed Salman, Van-Lan Dao, Alessandro Vittorio Papadopoulos, Saad Mubeen, and Thomas Nolte. Scheduling firm real-time applications on the edge with single-bit execution time prediction. In *2023 IEEE 26th International Symposium on Real-Time Distributed Computing (ISORC)*, pages 207–213. IEEE, 2023.

[99] Shaik Salman, Thomas Nolte, Alessandro Papadopoulos, and Saad Mubeen. Taming tardiness on parallel machines: Online scheduling with limited job information. Technical report, Mälardalen Real-Time Research Centre, Mälardalen University, October 2024.