Feedback Admission Control for Workflow Management Systems

Hashem Ali Ghazzawi, class one honours MEng

EngD

University of York

Computer Science

October 2015

Abstract

We propose a novel feedback admission control (FAC) algorithm based on control theory as a unified framework to improve the real-time scheduling (RTS) performance in industrial workflow management systems (WMSs). Our FAC algorithm is based on four main principles. First, it does not require the knowledge of RTS parameters of jobs prior to their arrival to the system for scheduling and processing. Second, it does not require a change of the scheduling architecture/policy in the industrial WMS which is a requirement in some industries including the one under consideration in this thesis. Third, we derive dynamic models for computing systems for the purpose of performance control. Finally, we apply established control laws to manage the trade-offs in meeting deadlines and increasing platform utilisation (classical RTS objectives).

The generality and efficiency of our proposed FAC algorithm are demonstrated by its application in three typical scheduling scenarios in industry. First, we tested our algorithm with simple tasks that are periodic and independent. For this application, we developed two FAC versions based on basic and advanced control laws to compare their performance with respect to the RTS objectives. Second, we added task dependencies as a scheduling constraint because they are witnessed in some industrial workloads. We evaluated our FAC algorithm against other baseline algorithms like the completion-ratio admission controller with respect to the RTS objectives. Third, we extended our FAC algorithm to support enterprise resource planning decisions in acquiring additional computing processors in real-time to further achieve the RTS objectives while constrained by industrial projects' financial budgets.

Contents

Abstract **3**

Table of Contents **5**

Acknowledgements **15**

Declaration **17**

1 Introduction **19**

2 Literature Survey **25**

2.1 The Case for Intelligent Admission Control Scheduling . . . 26

2.2 Intelligent Admission Control Scheduling: Modelling . . . 28

2.3 Intelligent Admission Control Scheduling: Decision-Making . . . 30

2.4 Intelligent Admission Control Scheduling: Control-Theoretic . . . 32

2.4.1 Fuzzy-Logic Control . . . 34

2.4.2 Feedback Control . . . 35

2.5 Task Dependencies as a Real-Time Scheduling Constraint . . . 37

2.6 Value-based Scheduling and Enterprise Resource Planning . . . 39

2.7 Summary . . . 41

3 Problem Formulation **43**

3.1 Basic Definitions . . . 43

3.1.1 Open-Loop Admission Control . . . 43

3.1.2 Feedback Admission Control . . . 43

3.1.3 HPC Cluster Processor Utilisation . . . 43

3.1.4 Slack Values . . . 43

3.1.5 RTS Objectives . . . 44

3.2 Research Hypothesis and Questions . . . 44

3.3 Thesis Structure . . . 45

4 A Novel Feedback Admission Control Scheduling Algorithm 47
4.1 Design Steps 48
4.1.1 Task Model 48
4.1.2 Scheduling Policy 50
4.1.3 Performance Metrics 51
4.1.4 System Model and Identification 51
4.1.5 Closed-Loop Scheduling Control 56
4.2 Design and Implementation of PID-based Feedback Admission Control 58
4.3 Design and Implementation of MPC-based Feedback Admission Control 61
4.4 Evaluations 66
4.4.1 Task Normalised Slack Values 67
4.4.2 Processor Utilisation 71
4.5 Summary 73

5 Handling Dependencies in Admission Control Scheduling 75
5.1 Task Model 76
5.2 Scheduling Policy 78
5.3 Performance Metrics 78
5.4 System Model and Identification 79
5.5 Closed-Loop Scheduling Control 79
5.5.1 A Novel MPC-based FAC Algorithm for Dependent Tasks 79
5.5.2 Baseline Admission Control Algorithms 81
5.6 Evaluations 82
5.6.1 Job Normalised Slack Values 83
5.6.2 Processor Utilisation 86
5.6.3 Job Killing Ratio and Wasted Processor Utilisation . . . 89
5.6.4 Profiling Killed Jobs 92
5.6.5 Enhancing AC_{Random} 94
5.7 Summary 95

6 A Preliminary Investigation of Feedback Admission Control for Cloud and ERP Applications 97
6.1 Task Model 98
6.2 Scheduling Policy 98
6.3 Performance Metrics 99
6.4 System Model and Identification 99
6.5 Closed-Loop Scheduling Control 99
6.5.1 A Novel MPC-based VAC Algorithm for Cloud and ERP Applications 100
6.6 Evaluations 102

6.6.1 Job Normalised Slack Values 103
6.6.2 Processor Utilisation 106
6.6.3 Project Budget 108
6.6.4 Jobs Killing Ratio and Wasted Processor Utilisation . . 110
6.6.5 System Credit 113
6.6.6 Trade-off Analysis 116
6.7 Summary 117

7 Conclusions and Further Work 119
7.1 Further Work 121

A Appendix A - Periodic and independent task-sets 123

B Appendix B - Aperiodic and dependent task-sets 125

C Appendix C - Aperiodic and dependent task-sets with ERP 127

References 131

List of Tables

4.1 Task model of generated tasks [42, 43]. For Data tasks, a Round-Robin selection criteria is implemented inside each range, and random switching criteria is implemented between the two ranges. This can be noticed in Figure 4.4 (top). For Design tasks, they have only one range and values are selected via a random selection criteria. 50
4.2 PID parameters tuning as depicted from Lu et al. in [75]. . . . 60
4.3 One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of ntS_{Design}. 71
4.4 One-way ANOVA test for the admission control experimental data of ntS_{Design}. 71
4.5 One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of PU. 73
4.6 One-way ANOVA test for the admission control experimental data of PU. 73

5.1 Task-set's RTS specifications [41]. 76
5.2 One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of njS. 86
5.3 One-way ANOVA test for the admission control experimental data of njS. 86
5.4 One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of PU. 88
5.5 One-way ANOVA test for the admission control experimental data of PU. Mark (*) indicates that the difference between the data-sets is almost insignificant. 88
5.6 Simulation time ($SimTime$) of all algorithms. $SimTime$ indicates how long each algorithm spent in scheduling and processing all arriving jobs. 89
5.7 One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of jKR and WPU. 91
5.8 One-way ANOVA test for the admission control experimental data of jKR and WPU. 91

6.1 One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of njS. 105
6.2 One-way ANOVA test for the admission control experimental data of njS. Mark (*) indicates that the difference between the data-sets is almost insignificant. 105
6.3 One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of PU. Mark (*) indicates that the difference between the data-sets is almost insignificant. 107
6.4 One-way ANOVA test for the admission control experimental data of PU. 107
6.5 One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of jKR and WPU. Mark (**) indicates that the data-sets are equal; when we ran the ANOVA tests in both Matlab and R programming environments, we obtained p-value = NaN. 112
6.6 One-way ANOVA test for the admission control experimental data of jKR and WPU. Mark (**) indicates that the data-sets are equal; when we ran the ANOVA tests in both Matlab and R programming environments, we obtained p-value = NaN. 112

A.1 Average values for OL, P-, PI-, PID- and MPC-based FAC algorithms with FCFS and EDF policies in a 10-processor HPC cluster with respect to Data & Design tasks ntS values and PU QoS levels. For more details on the algorithms, please refer to chapter 4. 123

B.1 Results summary of admission control algorithms performance comparison in dependent soft real-time tasks. Each algorithm is tested with FCFS and EDF in a 50-processor HPC cluster. Each performance metric is experimented with low (80%), full (100%) and high (120%) workloads. For more details on the algorithms, please refer to chapter 5. 126

C.1 Results summary of admission control algorithms performance in dependent soft real-time tasks with financial incentives and constraints. Each algorithm is tested with FCFS and EDF in a 50-processor HPC cluster. Each performance metric is experimented with low (80%), full (100%) and high (120%) workloads. For more details on the algorithms, please refer to chapter 6. 128

List of Figures

1.1 A block diagram of typical industrial workflow management systems. 19

1.2 A simplistic illustration of admission control. Percentages represent the HPC utilisation value per job. 21

4.1 A block diagram of a typical feedback control system. 47

4.2 (a) Design steps for feedback control real-time scheduling systems. (b) A block diagram of our feedback admission control algorithm. 49

4.3 A block diagram of the open-loop (OL) version of our industrial system with emphasis on the System Model components. The System Model block represents the system identification model of the scheduling and processing dynamics of our industrial system which the FAC block uses in its admission control decision-making process in Figure 4.2-(b). 51

4.4 Simulated system inputs for system identification modelling. We can notice for the Data tasks (top plot) the Round-Robin selection criteria within each range, and the random switching mechanism between the two ranges. 53

4.5 Simulated system outputs for system identification modelling. . 53

4.6 Our system identification model accuracy: measured minus simulated data. The reason the x-axis is not the full 200 hours is because in system identification we allocate one part of the measured data for the model estimation and another for model's accuracy validation [46]. 56

4.7 (a) A schematic representation of PID control. (b) Our novel PID-based feedback admission control algorithm where the System Model block is the System Identification model obtained from subsection 4.1.4. 58

4.8 (a) A schematic representation of MPC. (b) Our novel MPC-based feedback admission control algorithm where the System Model block is the System Identification model obtained from subsection 4.1.4.. 62

4.9 Design tasks normalised slack values. 68

4.10 MPC-based FAC algorithms with FCFS in a 10-processor HPC cluster. The values for $Data_{ntS}$ and $Design_{ntS}$ in the top plots are for the MPC-based FAC algorithms. 69
4.11 MPC-based FAC algorithms with EDF in a 10-processor HPC cluster. The values for $Data_{ntS}$ and $Design_{ntS}$ in the top plots are for the MPC-based FAC algorithms. 69
4.12 Processor utilisation. 72

5.1 Tasks' dependency patterns [15]. 76
5.2 An example for calculating deadline values for critical and non-critical tasks. The task path with thick arrows represent the critical tasks forming the job's critical path. 77
5.3 Tasks and jobs system identification model error: measured minus simulated data. The reason the x-axis is not the full 200 hours is because in system identification we allocate a small part of the measured data for the model estimation and the rest for model validation [46]. 80
5.4 A block diagram of our new, novel MPC-based FAC algorithm proposed for handling dependent real-time tasks referred to as AC_jAC_t due to the double Admission Control blocks. 81
5.5 A block representation of our baseline algorithms. 81
5.6 An illustration for admission decision-making of AC_{Random} . The jobs highlighted in italic are the ones which AC_{Random} can consider for killing because they have not started processing in the cluster. The other jobs i.e. $j1$ and $j2$ are outside the decision-making range for AC_{Rand} because they are partially i.e. some of their tasks are being processed in the cluster and hence removing those jobs can decrease PU and also cause WPU as both consequences are undesired. 82
5.7 Jobs normalised slack values box-plots of high workloads, 120%. 85
5.8 Processor utilisation time-plots of high workloads, 120%. 87
5.9 Jobs killing ratio time-plots of high workloads, 120%. Please note that we zoomed the time-plots to include the first 50 hours of the simulations because it is during this period the cluster was found to be fully utilised. 90
5.10 Wasted processor utilisation box-plots of high workloads, 120%. 90
5.11 Profiling jobs killing by the admission control algorithms with FCFS with respect to killed jobs' jC values and the time of rejection. 93
5.12 Profiling jobs killing by the admission control algorithms with EDF with respect to killed jobs' jC values and the time of rejection. 93

5.13 Enhanced AC_{Random} algorithm with FCFS & EDF and high workloads, 120%. 94

6.1 System identification model error: measured minus simulated data. The reason the x-axis is not the full 200 hours is because in system identification we allocate a small part of the measured data for the model estimation and the rest for model validation [46].. 100
6.2 A block diagram of our novel value-based admission control (VAC) algorithm. 101
6.3 Jobs normalised slack values box-plots in high workloads, 120%. 104
6.4 Processor utilisation time-plots in high workloads, 120%. 106
6.5 Project budget plots in high workloads, 120%. We zoomed-in to show the RTS dynamics during the cluster overload period. . . 109
6.6 Jobs killing ratio time-plots in high workloads, 120%. We zoomed-in to show the RTS dynamics during the cluster overload period. 111
6.7 Wasted processor utilisation box-plots in high workloads, 120%. 111
6.8 System credit plots in high workloads, 120%. We zoomed-in to show the RTS dynamics during the cluster overload period. . . 114
6.9 Trade-off analysis between multiple RoD rent prices and PoD in high workloads, 120%. 116

Acknowledgements

First and for most, to God, I thank for the strength and faith that keep us standing and for the hope that keeps us believing and succeeding in our life.

To my family:
I start with the chief, the late Ali Ghazzawi, all I can utter is I wish you were around, and I am proud belonging to you, dad. To the passion of this family, the centre of its warmth and its shelter, Sabah. Thank you for being around me when I need you, and thank you for all your prayers and blessings upon me, mum. To my brothers and sisters: Housam, Manal, Hadeel (and Mahmoud), Ahmad and Amr. Thank you for your exceptional friendship, I am the youngest of you all and I am ever proud belonging to you.

To University of York's Real-Time Systems Research Group:
I start with my amazing supervisors, Iain Bate and Leandro Soares Indrusiak, who by now know me more than anyone else in the world. You have shown me, many times, my strengths and weaknesses, and have always guided me to fruitful decisions. Your help will always stay with me because your help, not only being professionally great, but also personally resonating. Thank you both so much. To my internal assessor, Jon Timmis, thank you a great deal for your professional help and continuous support. To the rest of the research group, a great gratitude goes to the chiefs, Alan Burns and Andy Wellings, and a big thank you goes to the rest of this friendliest team ever.

To my seniors and colleagues in the industrial partner of this work. Thank you all for your support.

A big thank you to all my friends out there.

Yours sincerely,
Hashem (aka #em)

Declaration

This thesis has not previously been submitted in substance for any degree, and it is not concurrently submitted in candidature for any degree other than degree of Doctor of Engineering of the University of York. This thesis is the result of my own investigations, except where otherwise stated. Other sources are acknowledged by explicit references. Certain parts of this thesis have appeared in previously published papers specifically the following references (marked * for principal author):

- H. A. Ghazzawi. Scheduling Approaches for Large-Scale Complex Task Management. The Proceedings of the 2nd Large Scale Complex IT Systems (LSCITS) Postgraduate Workshop, pp. 60-66, 2010.
  Parts of this paper were used in chapter 1 and chapter 2 for motivating our research.

- H. A. Ghazzawi*, I. Bate and L. S. Indrusiak. A Control Theoretic Approach to Workflow Management. In Proceedings of 17th IEEE International Conference on Engineering of Complex Computer Systems, pp. 280-289, 2012.
  This paper forms one part of chapter 4 in introducing our novel feedback admission control algorithm.

- H. A. Ghazzawi*, I. Bate and L. S. Indrusiak. MPC vs. PID Controllers in Multi-processor Multi-Objective Real-Time Scheduling Systems. In Proceedings of the 2012 UK Electronics Forum, pp. 77-83, 2012.
  This paper forms another part of chapter 4 in comparing our basic and advanced feedback admission control algorithms.

- H. A. Ghazzawi. A Control-Theoretic Approach for Scheduling Soft Real-Time Tasks with Dependencies. Technical Report YCS-2014-495, University of York, Department of Computer Science, 2014.
  This paper forms chapter 5 in introducing how our novel feedback admission control framework addresses task dependencies as a scheduling constraint.

Chapter 1

Introduction

In this work, we are concerned with industrial workflow management systems (WMSs). The term *workflow management* refers to a business automation software to facilitate technology for decision-making and handling of IT projects with respect to organisational objectives such as high customer service rate [107]. Workflows can be computational fluid dynamics (CFD) simulation jobs that are admitted into the WMS for scheduling and processing. The workflows are processed in high performance computing (HPC) clusters due to their high performance and ability to process numerous jobs in parallel [90].

It is a common situation for industrial WMSs to be overloaded when processing CFD jobs due to their finite computational resources. Overload situations can incur longer queueing time for jobs which can lead to missing deadlines for some of those CFD jobs, which is undesired. Workload regulation is one answer to avoid deadline misses via admission control practices. Admission control is a mechanism for admitting or rejecting certain jobs which have arrived into the system for scheduling and processing. Admission decisions are practised with respect to classical real-time scheduling (RTS) objectives including minimising deadline misses and maximising platform (HPC processor) utilisation. Refer to Figure 1.1 for a general block diagram of industrial WMSs.

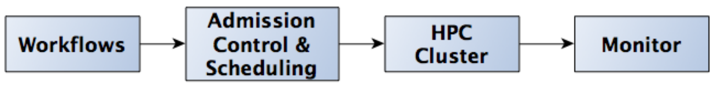

Figure 1.1: A block diagram of typical industrial workflow management systems.

Some industrial organisations practice admission control through a human system administrator. This can cause issues in two strands. Before we address them, we need to define what we mean by workflows, jobs and tasks. We define a workflow to be a bundle of N number of CFD jobs which can be parallel. Each job can, in turn, contain a number of dependent tasks. Dependency refers to the particular processing pattern of tasks whereby a task cannot start processing until it receives data from a predecessor task which have completed processing. Individual tasks are not parallel, each task can occupy one processor at a time and it is an indivisible unit requiring certain processing time.

First, administrators can and do mistakes in managing the trade-offs between the RTS objectives. In other words, system administrators may reject (kill) the wrong job or kill a job but at the wrong time. Managing the trade-offs between the RTS objectives can be challenging: "*if we admit most arriving jobs, we might increase processor utilisation (desired) at the expense of missing deadlines at cluster overload periods (undesired). Alternatively, if we kill most arriving jobs, we might avoid or minimise deadline misses (desired) at the expense of poorly utilising the HPC cluster (undesired)*".

We desire an automatic and intelligent admission control mechanism that regulates workloads with respect to managing the aforementioned trade-offs between the RTS objectives. For instance, an intelligent admission controller may kill certain newly arrived jobs due to its prediction that such jobs will miss their deadlines because the cluster is overloaded or near overload (due to queueing jobs). If such jobs are admitted, they will increase the number of system's deadline misses because of their own misses (undesired). They will also unnecessarily load the cluster and potentially have other jobs being delayed, hence potential increase of deadline misses (undesired).

Second, system administrators do not necessarily compute their admission decisions based on past, current and futuristic (predicted) states of the system. What they usually do is estimate if newly arrived jobs will meet/miss their deadlines based on real-time snapshots of system states representing current processing capacity and performance. This practice is not effective because we want to evaluate the current effects and predict the future effects of the admitted jobs on the RTS objectives. Snapshots cannot produce such predictions and system administrators usually use such snapshots for instantaneous admission control decisions regardless of past performance and future effects.

Assume we have Job0 and Job1 that are composed of independent tasks and utilise 50% and 75% of the cluster, respectively. Also, assume Job0 is *currently* inside the cluster being processed. From an admission control point of view, if Job1 arrives, then system snapshots can reveal to us that half of the cluster is free and Job1 will occupy 75% if it is admitted *now* causing cluster overload at 125% (Job0(50%) + Job1(75%)). The system administrator can then decide to, for instance, admit 50% of Job1 (i.e. a number of tasks belonging to Job1) to (1) fully utilise the cluster, (2) start processing Job1 to achieve its deadline, and (3) avoid overload, see Figure 1.2.

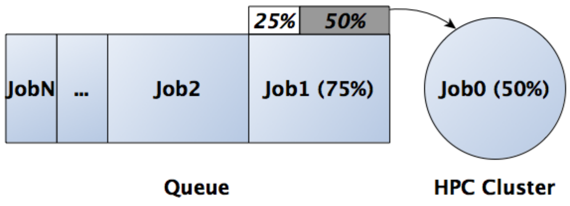

Figure 1.2: A simplistic illustration of admission control. Percentages represent the HPC utilisation value per job.

For this particular example, this admission decision is desired with respect to increasing the cluster processor utilisation. However, this can be challenging when dealing with tasks dependencies within jobs. In the presence of dependencies, a system administrator cannot admit 50% of Job1 due to the particular processing pattern within the job being enforced by its internal (task) dependencies. The system administrator then needs to predict Job1's utilisation of the cluster over time. So, at least a high-level prediction of the cluster's processing capacity is obtained with respect to the admission of future jobs. The system administrator may not know future jobs' RTS parameters including arrival and computation time nor deadline values.

We can review the issues of admission control scheduling which this thesis addresses in three strands. Our research issues are not only seen in general WMSs, but also form part of a specific industrial WMS which this thesis uses as a case-study [40]. Our review here will briefly include the general trends of admission control scheduling approaches in the literature. However, a detailed review can be found in chapter 2.

- **On-line admission of jobs.**

This refers to the on-line admission of jobs into HPC clusters with respect to managing the trade-offs between the RTS objectives. There are two issues here. **First**, the lack of knowledge of jobs' RTS parameters (e.g. release rate, deadline and computation time values) apriori i.e. prior to their arrival to the system for scheduling and processing. Most available admission control scheduling approaches in the literature assume some, if not exact, knowledge of jobs' RTS parameters apriori as part of their on-line admission control scheduling algorithms. In this thesis, we assume such information is unknown apriori.

Second, the admission controller should be independent, in its decision-making logic, from the scheduling policy being adopted in the WMS. There are related works in the literature that have integrated the design of the admission controller with the scheduling policy. However, a requirement in an increasing number of industrial WMSs is to separate the two in the design phase. This requirement stems from the need, in some industries, to have their admission controller flexible enough to continue practising its admission decision-making logic despite changing the scheduling policy in the WMS.

Due to the two aforementioned issues, we derive the first research objective for this thesis: "*to propose an intelligent admission controller that predicts the effects of arriving jobs on the RTS objectives via a model-predictive algorithm without prior knowledge of jobs' RTS parameters*". **Our main contribution in this thesis** is introducing a novel admission control algorithm based on comparing arriving jobs' computation time values against the predicted value of the system's processing capacity (in computation time units).

With this novel approach, we do not require prior knowledge of jobs' RTS parameters because we can know jobs' computation time values the instant they arrive to the system. Model-predictive admission control can manage the trade-offs between RTS objectives by using past and current system performance (states). This enables the admission controller to predict how far can the system continue processing jobs without, for instance, missing deadlines. This is addressed, reviewed and evaluated in chapter 4.

- **Deal with task dependencies as a scheduling constraint.**

Task dependencies force a scheduling issue for jobs. An admission controller can admit and process one part of a job (a number of dependent tasks) and then, later, reject another part of the same job. In this thesis, we assume the time the cluster spent processing the first part of the job will go to waste. This is because end-users require jobs to be completely processed. Most related works in the literature have addressed the dependencies issue assuming jobs RTS parameters (including dependency patterns) can be known apriori. However, the literature lacks approaches for dealing with dependencies via on-line admission control scheduling without prior knowledge of jobs' RTS parameters.

From this issue, we derive a second research objective for this thesis: "*to propose an admission controller that computes an admission decision for arriving jobs (unknown apriori) while respecting their task dependencies in minimising such wasted processor utilisation while managing the trade-offs between the RTS objectives*". This is addressed, reviewed and evaluated in chapter 5.

- **Deal with jobs' financial values as incentives and constraints.**

Some jobs might not be financially plausible for admission into the system because the value earned from processing such a job can be less than the cost of processing it locally or in, for instance, the cloud. The trade-off here is we want to process as many jobs as possible to earn more financial rewards but we are limited with the project budget which allows us renting or buying additional computing processors. Acquiring additional processors can help the local cluster, which is usually finitely resourced, for processing more jobs in order to earn extra rewards.

Most related works in the literature have dealt with jobs' values as incentives for meeting deadlines in order to acquire those rewards. However, the literature lacks approaches for dealing with both jobs' values and project budget as real-time incentives and constraints, respectively, to acquire additional computing processors. We refer to this mechanism as enterprise resource planning (ERP). From this issue, we derive our third research objective for this thesis: "*to propose an admission controller that can evaluate, in real-time, whether it is financially plausible to acquire additional computing processors to support the local cluster in processing more jobs while handling task dependencies and managing the trade-offs between the RTS objectives*". This is addressed, reviewed and evaluated in chapter 6.

Chapter 2

Literature Survey

The concept of real-time systems adopted in this work is “a system that is required to react to stimuli from the environment (including the passage of physical time) within time intervals dictated by the environment” [62]. “The correct behaviour of such systems depends not only on computational results, but also on the time at which the results are produced” [109]. In real-time systems, jobs can be admitted or killed on-line based on predictions whether they will meet their deadline or not. The algorithm responsible for testing the schedulability of a new job in a system is normally referred to as admission control [84]. Admission control-based algorithms form a major paradigm in RTS and a rich body of results are found in the literature where their central approach is based on on-line admission control and planning and thus applicable in resource-constrained environments [77].

In this chapter, we review related work that motivated the case for intelligent admission control scheduling in section 2.1. We then review the modelling and decision-making aspects of admission control scheduling in relevant works in the literature where we motivate, in a high-level, the case for control-theoretic approaches, in section 2.2 and section 2.3, respectively. As for modelling-related admission control scheduling, we will review how some related works have integrated the design of admission control with the adopted scheduling policy, which is undesired in our work here.

We will also see how some works have assumed certain RTS task-model assumptions that set their work apart form ours such as **the assumptions for periodic jobs and the ability to control jobs’ release rates (periods) which are undesired and infeasible in the WMSs we are concerned with in this work**. This is because end-users’ submission rate varies depending on the industrial projects they are actually involved with. We review the control-theoretic solutions in large and we motivate the case for feedback control for our research work in this thesis in section 2.4. Finally, we review related work on task dependencies and value-based scheduling constraints in section 2.5 and section 2.6, respectively.

2.1 The Case for Intelligent Admission Control Scheduling

For real-time systems, there are static and dynamic scheduling algorithms [40]. On the one hand, the static ones assume full and exact knowledge of task-sets and their associated RTS parameters such as deadline and computation time values apriori. Rate-monotonic (RM) is one exemplar in RTS systems which prioritises jobs of the highest period values i.e. their release rate, this can aid the system developer to categorise the RTS system **apriori** [59, 60]. Lauzac et al. mention in [64] that implementing RM in admission control scheduling with respect to processor utilisation was first proposed by Liu and Layland in [69]. There are two issues with RM. **First**, it is difficult in some industrial systems to predict the RTS parameters of the incoming jobs for scheduling and processing. **Second**, RM's schedulability bound is less than 100%, in fact it is 69% [69], and one of our RTS objectives is to maximise HPC processor utilisation.

On the other hand, dynamic algorithms have been developed for resource (in)sufficient systems for guaranteeing jobs to be schedulable despite their dynamic arrivals to the system. One of the popular dynamic RTS policies is earliest-deadline-first (EDF) [18, 105]. EDF uses jobs' deadline values as the priority level; the earlier the deadline the higher the priority. EDF's schedulability bound is 100% which is, indeed, always desired. Yet, one major downfall of EDF is the lack of certainty in determining which jobs will miss their deadlines in the case of cluster overload situations [60]. Lu argues in [73] that EDF lacks performance in resource-constrained systems and Maggio et al. in [79] mention that EDF decreases in performance when the processor is overloaded. This is because EDF correlates priority levels with deadlines where there can be situations when a new job arrives all queueing (previous) jobs miss their deadlines [72].

EDF has also been implemented in admission control scheduling algorithms for real-time systems [110]. However, such admission control scheduling algorithms in the literature are developed for real-time communication and networking systems to support continuous traffic characterised by peak and average rates [36, 68]. Such systems assume periodic jobs in their workload models where they can control the release rate (periodicity) of jobs on-line in order to achieve schedulable solutions for incoming/trafficking jobs. Our research scope in

this thesis sets-apart from such an assumption in which we assume jobs are aperiodic and their periods are unknown prior to their arrival and therefore cannot be considered as an admission control variable.
As for resource insufficient systems, the Spring algorithm “can dynamically guarantee incoming jobs via on-line admission control and planning and thus it is applicable in resource insufficient environments” [11]. “The Spring scheduling algorithm is a dynamic, on-line solution that constructs guaranteed schedules based on deadlines, resources, precedence constraints, values, etc. The scheduler can be utilized off-line or on-line” [91].

However, the issue with all of the aforementioned scheduling algorithms is that they are open-loop (OL) which can be criticised in three strands [73]. **First**, once the scheduling algorithm is designed, implemented and launched into the system, it cannot adapt towards varying system states that were unforeseen during the design phase of the scheduler [105].

This issue is manifested in industrial WMSs where cluster overload situations might occur due to sudden or urgent business requirements that forced CFD engineers to flood the organisation’s HPC clusters with CFD jobs. **Second**, OL algorithms perform well in RTS systems which their workload can be modelled accurately, but consequently perform poorly in systems where we cannot obtain accurate workload models [73]. **Finally**, OL algorithms are usually built with worst-case workload models if accurate models are unavailable. In such cases, cluster under-utilisation is inevitable due to the pessimistic estimation of workloads.

Traditional real-time scheduling approaches were concerned with avoidance of undesirable effects such as overload and deadline misses. Adaptive real-time systems including on-line admission control scheduling are designed to adapt towards such effects dynamically with respect to the RTS objectives [73]. We desire intelligent admission control scheduling that can adapt in case of undesired situations like cluster overload in order to avoid undesired consequences such as deadline misses. Malrait et al. argued the need for adaptive algorithms replacing system administrators for automatically and intelligently regulating workloads in dynamic RTS systems instead of physical, manual tuning of systems’ RTS parameters [81]. We can see this trend of arguments in, for example, [6, 80, 82].

Although the literature supports adopting intelligent admission control scheduling for dynamic RTS systems, there are two main issues with related works that set them apart form our research scope. **First**, we see some works in the literature that have integrated the design of both of the admission controller and scheduling policy together. For instance, Peha in [98] claim that "... because scheduling is entirely independent of the admission control algorithm... This paper presents an approach in which admission control and scheduling are integrated". This goes against our desire in developing an intelligent admission control algorithm that supports the system in managing the trade-offs between its RTS objectives regardless the adopted scheduling policy.

Second, the assumption of a pre-emptive task-set model such as Quan and Chung's admission control work for real-time services under EDF in [99]. We refer to pre-emption in its classical RTS meaning where a task being processed inside a processor is pre-empted by another queueing task. Quan and Chung's work also includes the first issue we addressed earlier where they have inter-linked the design of admission control and scheduling together.

After surveying the literature and in addition to traditional RTS approaches, there exists two major aspects in the field of intelligent admission control scheduling: modelling (and assumptions on task-sets and processors etc.) and decision-making. These two aspects will be reviewed in the following two sections.

2.2 Intelligent Admission Control Scheduling: Modelling

One realm of intelligent admission control scheduling is artificial intelligence, in particular neural networks (NN) and genetic algorithms (GA). Using NN and GA was explored in the literature, but the common theme of such approaches are infeasible in the WMSs this thesis is concerned with. The reason for this is due to their modelling assumptions on:

- Independent task-sets and a single-processor real-time system [45].
- Knowing certain jobs' RTS parameters apriori such as the number of jobs per workflow [25, 26, 114].
- Pre-emptive RTS environments and enabling jobs migration across the cluster [23, 52, 83, 102, 112].

In addition to artificial intelligence, Ferrari in [35] advocates for admission control scheduling with the use of realistic and advanced system state models because simplistic models make the scheduling task more difficult in meeting deadlines given sudden and varying workload situations (models). However, Ferrari's work was concerned with real-time communication systems with the assumption of periodic jobs. Although it is ideal to work with realistic workload/system models, it is rather a challenging task to obtain such models. We desire a modelling technique that can dynamically generate system models with respect to varying workloads and processing capacity so the admission controller can use for its prediction and decision-making tasks.

The three aforementioned bullet-points concerning related works with independent task-sets, prior knowledge of jobs RTS parameters and pre-emptive task-sets have been addressed in this work by adopting a synthetic workload generator. The generator is the work of Burkmisher et al. in [16] and is validated with respect to our assumptions. It includes typical industrial task dependency patters and does not provide knowledge of jobs parameters to the system (admission controller) prior to their arrival to the system for scheduling and processing.

Cappanera et al. in [20] are in favour for obtaining system models that help the on-line admission controller in its feedback decision-making process. However, they assume knowledge of the arrival curve of incoming jobs. Also, Vacca et al. in [117] propose a feedback admission control algorithm for dynamic systems with jobs of varying arrival and computation time values. They use EDF scheduling as long as resources are sufficient. However, their work sets-apart from ours through their feedback mechanism because they adjust jobs' periods to be optimised with respect to meeting their deadlines.

Hyman et al. in [53, 54] also assume, for a broadband switching real-time system, the knowledge of calls' (jobs) arrival rates which they expect jobs to follow a certain arrival pattern. Jamin et al. in [56] propose a model-predictive admission control algorithm in which they do not use workload models per se. Instead, they utilise a number of equations each time a new job arrives to the system. These equations can, in real-time, predict whether the new job will meet its deadline or not. However, their admission control variable is jobs' periods. A similar trend of model-predictive admission control scheduling is also seen in Yang and Lu's work in [118] where they assume periodic jobs too.

As far as modelling of computing systems is concerned, we have queueing theory which has been used in RTS systems [100]. On the one hand, "in computing systems, queuing theory will not be enough when investigating admission control mechanisms. The queuing theoretic methods usually assume stable systems working in the non-overloaded region. Further, there are no mathematical tools in queuing theory for developing optimal admission control mechanisms. Instead, control theory may be used when investigating control mechanisms for queuing systems. However, queuing systems are both stochastic and non-linear, which introduces a number of modelling problems. These problems may be solved by finding mathematical approximations that mimics the behaviour of the system" [100]. Statistical modelling approaches such as Bayesian-based models require prior distribution of any unknown parameters.

We desire a model-predictive admission controller that copes with unknown events e.g. sudden cluster overload and arrival of unknown jobs without any prior knowledge of their RTS parameters. Control theory includes a modelling technique known as system identification which is initially done off-line and apriori using either (1) OL simulation of the system input(s) to record correlations (system dynamics) with system output(s), (2) back-logs of historical and recorded system dynamics (if available), or (3) system experts' opinion for simulating system dynamics. The admission controller uses the system identification model for:

- Constructing mathematical models approximating the real-time dynamics between system parameters (inputs and outputs) [92, 46].
- Capturing any sudden events in the system e.g. jobs arrivals and cluster overload etc. so it can prevent quality-of-service (QoS) violations via admission decisions.

2.3 Intelligent Admission Control Scheduling: Decision-Making

For industries that produce large-scale simulation jobs requiring heavy computations, there exists commercial products that are advertised to regulate these simulations into the HPC cluster. For example, LSF practices on-line admission control decision-making based on advanced reservation protocol of which it is designed to predict the performance (behaviour) of HPC clusters. LSF is a queueing system that implements reservation. However, there are limitations associated with this product such as under-utilisation [108]. This stems from the FCFS scheduling policy LSF implements.

Another example of commercially available HPC scheduling products is SynfiniWay, which is advertised to be platform-independent allowing its users to utilise their diverse and heterogeneous HPC platforms to its virtualised IT algorithm. It offers admission control and data handling via meta-scheduling decision-making capability. It is not the scope of this work to discuss SynfiniWay elements in details, analysis can be found in [113]. However, the limitations of SynfiniWay in industrial systems stem from the fact that it uses multiple servers to decide on workflow admissions. This can be time consuming because each SynfiniWay "neighbourhood" has its dedicated server monitoring the IT algorithm.

Another industrial example of HPC workflow scheduling is Oracle Grid engines [93] which (is run by Univa Corporations via its commercial WMS grid software [116]) practices admission control decision-making on-the-run by introducing a time-out for each session. Univa software is based on a pre-emptive RTS environment where jobs migration across the grid (cluster) is practised. Sessions here include a request made by the end-user to submit a job into the system domain. If it exceeds more than, for instance, a day, then the session is timed-out (killed) and the end-user will then need to re-submit the job again. This will avoid end-users submitting excessive jobs.

Handling rejected (killed) jobs is not present in this thesis. Currently in our industrial WMS, when a particular task causes the scheduling performance to degrade, the system administrator manually *kills* the whole job that contains this task. This is a serious flaw in the industrial system. Yet, some works in the literature have also assumed this admission decision-making mechanism. For instance, Bestavros and Nagy in [13] practice the killing of jobs where such jobs go to a *sink* with no re-admission mechanism. Their admission control decision-making process is based on feedback mechanism of current demand on system resources with respect to the RTS objectives which are high hit ratio (successful admission of jobs that have met their deadlines) and low killing ratio. However, their jobs are generated off-line and known apriori where the *varying/dynamic* part is jobs arrival time.

Univa's admission control and others in the literature can be considered a real-time admission control algorithm which prevents cluster overload situations by rejecting a certain number of jobs (sessions). Whereas there is another approach known as *fair-share* which is concerned with regulating the workloads as they arrive across the HPC cluster in a particular "fair" manner. The prime objective of fair-share decision-making mechanism, known as FairShare, is to achieve fair scheduling in regards to processor utilisation [61]. The issue is "it is concerned with the relationship between processes and users" [27]. It encourages users to spread their load on the machine [61]. In other words, it achieves its "fair" admission control decisions with respect to end-users by "fairly" regulating the workload across the cluster. However, design and production industrial IT systems "do not require interactive performance and spread-out their submissions of work" [17], and our industrial system is such a WMS type.

The assumption of pre-emption is pivotal in FairShare decision-making [61]. Implementing FairShare scheduling in a non-pre-emptive RTS system can lead to undesirable consequences such as end-users attaining higher share of their allocated segment of the cluster [17]. There is also multi-level feedback queues as an RTS heuristics for balancing jobs' response times in order to achieve FairShare scheduling [7]. However, it is also based on the pre-emption RTS assumption in its jobs execution model.

2.4 Intelligent Admission Control Scheduling: Control-Theoretic

Using control theory for improving RTS performance appeared only in the last decade [81], "while the concept of automated operations has existed for longer" [87]. As we have seen in the previous sections thus far that some if not all RTS and admission control approaches assume knowledge of system workload and service capacity apriori [73]. This can potentially cause poor:

- Prediction of scheduling performance in computing systems.
- Handling of dependencies.
- Management of trade-offs in multi-objective RTS optimisation.

This has been the motivation in the RTS community for alternative approaches that analyse and describe the aggregate behaviour of real-time systems [105].

One potential answer is the recent control-theoretic admission control which uses feedback mechanism to monitor the capacity of the cluster with respect to the QoS levels [30, 44]. The admission controller forces certain actions to regulate the workload for efficient RTS performance with respect to agreed-on objectives. This approach enables us handling the coupling between RTS systems components dynamically via feedback control [29]. This is achieved by defining performance metrics e.g. utilisation for transient response which can be mapped to dynamic response specifications of control systems (theory) [76]. Traditional approaches are concerned with statically assured avoidance of undesirable effects such as deadline misses and utilisation overload [78].

Adopting control theory provides optimised data/decisions that need to be translated into physical actions. This is necessary in order for the system to meet its timing constraints and QoS requirements. In physical systems, an actuator is known to be the responsible component that performs this task. In software systems, we have the notion of admission control, which limits the number of jobs being admitted into the system requiring computing processor utilisation [3]. Control theory helps overcoming the issues with OL RTS approaches in unpredictable environments where their system models cannot be modelled/obtained accurately. We can briefly list the roles of control theory in improving the performance of OL scheduling schemes from [74].

- First, it contributes to derive dynamic models of computing systems for applying performance control which is a pre-requisite for admission controller design [46, 92]. This can be achieved in the realm of controller design theory by a black-box modelling technique, system identification [71]. This is important when we desire to add more service qualities, in other words dealing with a multi-objective optimisation problem. This is more complex than having one optimisation goal because the admission controller needs to map each of its admission decisions with respect to the objectives simultaneously, chapter 4 further explains this. This relates to the **modelling** aspect of intelligent admission control scheduling.

- Second, it provides performance control for novel scheduling architectures via feedback on QoS levels, which is important for systems to meet qualities with precision and speed. In a dynamic and unpredictable real-time system, it is beneficial for the admission controller to be aware of the varying system states [105]. The reason this is beneficial is because in order to achieve optimised performance, we need to assure that the system reacts appropriately to different scenarios. If, for instance, the system is overloaded with queueing jobs and the admission controller predicts the deadline miss ratio is going beyond the QoS level. Then this will result in an admission decision like rejecting certain jobs arrived into the system to avoid violating a specific QoS level. This relates to the **decision-making** aspect of intelligent admission control scheduling.

A number of related works in the literature applied control theory for having the admission controller aware of the current state of the system. The reason for this is to optimise the decisions the admission controller makes to suit the changes occurred in the system behaviour. In Buttazzo and Abeni's work [19], they called their approach to be elastic. In other words, referring to adaptivity and resilience to sudden changes in system behaviour and/or QoS requirements. Buttazzo and Abeni utilised feedback control laws to monitor the states of the system so the scheduler can consider admission decisions excluding the busy components.

If every newly arrived job is sent to the cluster straight away, then an issue can arise here. The new job might occupy a fast HPC processor that potentially could serve a larger queueing job (with respect to processor cycles). This can further impose unnecessary queueing time for the larger job to wait for the fast HPC processor to become free. Their work and others such as [70] show the lack of adaptivity to variance in classical (traditional) RTS approaches. Such approaches are designed to the specific characteristics of digital control systems instead of general adaptive real-time computing systems. Other related works proposed adaptive processor scheduling algorithms or QoS management architectures for computing systems such as multimedia and communication systems [4, 65, 101, 111].

We can generalise that some approaches can improve the RTS performance by controlling i.e. real-time assignments of optimal deadline and release time/rate values via feedback monitoring and control e.g. [76, 96]. A clear motivation for feedback-based admission control scheduling can be found in numerous researches e.g. [9, 74, 105].

Feedback control is one branch of control-theoretic RTS, whereas fuzzy-logic control forms another branch of it. The next sub-sections review related fuzzy-logic and feedback control works in the literature.

2.4.1 Fuzzy-Logic Control

The literature offers some research works that can be relevant to the RTS scope of industrial WMSs. Fahmy's work in [33] offers a fuzzy-based GA algorithm for scheduling non-periodic jobs. However, his work supports single-processor systems only. Hongjun et al.'s work in [50] supports non-periodic jobs for RTS operating systems lacking jobs' parameters apriori such as AMD and Intel. However, their work is based on pre-emptive RTS environments. A similar trend of related fuzzy-logic admission control scheduling works assume periodic jobs, jobs migration across the cluster, pre-emptive RTS environment and knowledge of jobs' RTS parameters apriori are found in the literature.

The main reason this control-theoretic approach is not suitable for the type of industrial WMSs this thesis is concerned with is because fuzzy logic-based control systems assume full knowledge of the system dynamics apriori. For instance, a fuzzy logic-based washing machine can dynamically decide how clean/dirty the clothes load is based on pre-knowledge of what *clean* is and what *dirty* is based on prior models being fed to the fuzzy logic controller. Once the controller decides the level of *dirtiness*, it can then choose a suitable washing-level for that particular load of clothes. This is infeasible in the WMSs we are concerned with because we lack accurate models of workloads. If workloads can be accurately modelled, then the literature offers a rich body of approaches of traditional admission control scheduling algorithms.

Feedback control theory, however, can work satisfactorily with simplistic system models derived analytically. In the case of difficulty in deriving such models analytically, which is the case with most real-time computing systems [46, 73], there are system identification techniques that supports the feedback controller with dynamic models of the system states. The next sub-section reviews feedback control-based admission control scheduling approaches and motivates it for our work carried-out in this thesis.

2.4.2 Feedback Control

A classical control algorithm known as proportional-integral-derivative (PID) has been used in uni-processor real-time systems to reduce deadline miss ratio [75]. PID is a basic feedback control to achieve admission control scheduling objectives in real-time systems [92]. However, when common characteristics arise such as large time delays, non-linearity, high-order dynamics or multi-processor cases, PID lacks performance improvement as it is linear and in particular symmetric [37].

In [76], two PID controllers were implemented by Lu et al. to meet two RTS objectives; maximising processor utilisation and minimising deadline miss ratio. Lu et al.'s attempt in addressing RTS multi-objective optimisation has lead to significant results in their single-processor case-study. In the literature, we can see how PID is argued, in theory, to be used for multi-objective optimisation [38], where a mechanical industrial experimentation is illustrated in [97]. However, later on when multi-processor scenarios were examined by Lu et al., model-predictive-control (MPC) has offered promising results for the same RTS multi-objective optimisation problem [78]. This very work of Lu et al. has inspired us adopting MPC supporting admission control decisions for RTS [42] as MPC-based admission control was suggested in [78].

MPC is an advanced feedback control algorithm for controlling real-time dynamic systems. It is popular in industries that heavily rely on process control - a survey of MPC applications in industry is provided by Nicolaou in [88]. In the literature, MPC has been advocated for controlling systems that exhibit non-linear behaviour and contain multi-input-multi-output (MIMO) structure. A survey of feedback control in software services is compiled by Abdelzaher et al. in [1]. Most, if not all, computing systems exhibit non-linear behaviour [46], and our industrial WMS has a MIMO structure since both the number of system inputs and outputs exceed one, more details on this point can be found in chapter 4.

Lu et al. chose MPC for their work in minimising the difference between the desired processor utilisation values of m processors (set-points) and the actual values monitored with time [78]. The control variable was periodicity, which controls the release rate of jobs. For control jobs, there are two approaches for controlling processor utilisation, either by manipulating jobs release rates [78] or computation time values [48]. Our approach is based on the latter for jobs admission because controlling jobs release rate values is undesired in our industrial system as the distribution of submitting workflows into the HPC cluster is and will not be controlled.

Another advanced feedback control approach is linear quadratic regulator (LQR) control algorithm for multi-objective RTS optimisation. Similarly to MPC, LQR captures the dynamics of a system by a set of linear differential equations that can be derived via system identification [71] e.g. the work of Diao et al. in [28]. MPC differs from LQR by allowing explicit optimisation constraints on both the system outputs and control signals (variables) [10]. Both MPC and LQR handle optimisation via their quadratic optimisation approach [104]. A quadratic programming (QP) problem has an objective which is a quadratic function of the control signal(s), and constraints which are all linear functions of the control signal.

Diao et al. chose LQR feedback admission control algorithm in optimising processor and memory utilisations of an Apache server via controlling the maximum number of requests made to the server, and the duration of keeping a request alive or queueing to be served [28]. However, it is not desired, in industrial WMSs we are concerned with in this thesis, to force a “cap” on admitting jobs in the case of receiving one with a high priority but exceeds the maximum number of jobs specified by the admission controller. Cherkasova and Phaal have also used “cap” feedback admission control logic in [24].

In Lu et al.’s feedback admission control approaches in [76, 78], jobs killing (removal from the system) is not practised, but rather, assigning a futuristic, optimal release time value for jobs. We assume the assignment of future/optimal release time values to be infeasible in the industrial WMSs this thesis is concerned with. Similarly to Lu et al.’s feedback admission control approach, there is Park and Humphrey’s notion of Scheduling Credits which are assigned for the admission of prioritised jobs [96]. However, Park and Humphrey used control laws to optimise the values of jobs’ periods with respect to high processor utilisation and minimum deadline misses which requires knowledge of jobs’ periods apriori.

Similarly to Lu’s feedback admission control work, Heo et al. developed a distributed performance optimisation service in a 3-tier Web server farm [49]. The RTS objective was to minimise the energy cost function of 18 machines, a HPC test-bed. The admission control logic consisted of regulating the rate for the incoming back-up requests which is infeasible in our industrial system. On the other hand, Robertsson et al.'s work does not control the arrival rate of jobs, but instead they assume prior knowledge of the arrivals where "the mean arrival rate will, in practice, vary with time and has to be estimated, but it is in this paper assumed to be constant and known" [100].

Finally, feedback admission control scheduling can be a promising candidate for our industrial WMS via its capabilities in on-line (1) system modelling (model-predictive) of the effects of newly arrived jobs' computation time values on the system's processing capacity and (2) decision-making process in comparing newly arrived jobs' computation time values with the system's processing capacity. Despite this, there are two industrial scheduling constraints that have been addressed in the literature for dynamical real-time computing systems. They are tasks' dependencies and job's financial values for admission. The next two sections will review these constraints, respectively.

2.5 Task Dependencies as a Real-Time Scheduling Constraint

There are traditional RTS approaches that have been studied for addressing the dependency constraint or issue between real-time task-sets. Saksena and Hong applied Critical Scaling Factor [66] for RM scheduling of periodic jobs in [103], and Abdelzaher and Shin investigated the Branch and Bound algorithm for, also, periodic task-sets [2].

The cited algorithms find a schedule using jobs periodicity as a parameter or condition. They can lack performance in dealing with jobs only upon their arrival which do not, necessarily, follow a particular periodicity (release pattern). This motivated our work in overcoming this issue by dealing with jobs upon arrival to compute an admission decision based on the jobs computation time values regardless of periodicity.

The work of Saksena utilised deadline assignments for satisfying end-to-end job constraints off-line i.e. before launching the scheduler into the system. This approach requires knowledge of system's workload parameters and computing capacity apriori. The practice of designing optimal deadlines for meeting RTS constraints based on workload models [103] and jobs arrival times [2] is a common trend. Overall, the cited works in the literature take metrics like computation & release times and precedent constraints (task dependencies) etc. as part of their scheduling algorithms.

In the literature, there are also dynamic scheduling algorithms addressing the issue of dependencies. However, some approaches assume a different RTS problem from our industrial system's. For instance, in the work of Simon et al. in [106], they take task dependencies to be in regards to delays i.e. real-time response time, but not to dependencies between the real-time tasks themselves i.e. their processing patterns. Also, in the work of Feitelson et al. in [34], they deal with the jitter of dependent real-time tasks. However, task dependencies have not been dealt with as a scheduling constraint per se, but network delay and jitter was the constraint, instead.

Cervin addressed data dependencies between real-time tasks to being a scheduling constraint that can degrade the controller performance in meeting the desired RTS objectives, hence scheduling performance degrades too [21]. According to his feedback control analysis, the computation time of tasks "may vary randomly due to data dependencies...". In Cervin's feedback control algorithm, "the timing description is somewhat limited in that it cannot handle dependencies between periods".

The literature also offers feedback control-related works that have addressed the issue of task dependencies in RTS systems. Yuan et al.'s research was concerned with the scheduling of control systems. Their problem statement is "the problem we consider here is to find the execution order of n periodic tasks with task dependencies" [119]. We notice there are two RTS assumptions that make their work sets-apart from ours. The first assumption is they deal with a particular number of tasks known apriori. The second is they deal with periodic task-sets.

However, Yuan has furthered the work to include handling aperiodic tasks in [120] where his research problem is concerned with scheduling the control tasks of a multi-robot system. We can consider the robots' arms to be our HPC processors in order to bring Yuan's work and ours closer. His work is based on adjusting the frequency of tasks so deadlines can be met. Otherwise, an admission controller is then utilised for rejecting least priority oncoming tasks and terminating the processing of such tasks too. Changing tasks (or jobs) frequency in the cluster is not desired in our industrial system, the system administrator does not have the option for compromising the quality of the CFD jobs computational results by reducing their frequency rate values for the sake of meeting deadlines. Additionally, Yuan's admission controller assumes a pre-emptive RTS environment as well.

If we compare Yuan's and other RTS systems in the literature with the WMSs we are concerned with, we find that in our industrial system, if a job is admitted, then it is expected to run for its complete duration. Therefore, we require a feedback admission control algorithm to meet our RTS objectives while respecting task dependencies and *without* compromising any of the release and computation time values.

One part of our research will be focussed on designing an algorithm that deals with task dependencies as a scheduling constraint, refer to chapter 5 for more details on this topic.

2.6 Value-based Scheduling and Enterprise Resource Planning

Value-based scheduling is concerned with the rewards (financial values) earned after scheduling and processing jobs on time. Lai in [63] showed that adopting value-based scheduling will result in a different scheduling architecture such as market-based. This can be undesired in industries not wishing for changing their current existing scheduling architecture including our industrial system.

As for job value assignment, literature shows that having a dynamic value (rewards) is better than a fixed one for improved market efficiency [63]. There are different dynamic models, e.g. [22, 67], for designing (job) values that, indeed, affect the scheduling performance with respect to earning maximum values [17]. When the value is modelled, then the admission controller is expected to earn the maximum value return for the whole workload [55, 58].

Our scheduling problem differs from mainstream literature in which we cannot afford, in the industrial system, to change the scheduling structure and policy in order to accommodate value-based scheduling architecture. The reason behind this is because changing the scheduling policy would lead to changing the scheduling system architecture [14]. This is undesired in our industrial system and many other industrial organisations would prefer adding a software layer that would regulate the workload into the processing cluster which was an earlier motivation for our feedback admission control algorithm. Therefore, it is the responsibility of our proposed admission controller to earn this maximum value (reward) by carefully admitting jobs into the cluster and potentially make automatic decisions for acquiring additional resources if found necessary via purchasing new processors and renting from the cloud.

Within the European EU/IST FP6 Network of Excellence Artist2 on Embedded System Design, a roadmap on Control of Real-Time Computing Systems has been published in the last decade. It specifically addresses the value-based scheduling within, of course, the control-theoretic perspective. The road map states that "an interesting but still largely unexplored approach is to instead use value-based or direct feedback scheduling. Here, the idea is to base the decision of which task to execute on an instantaneous cost function. This cost function should grow the longer the control loop executes in open loop and decrease when a control action is issued. The instantaneous cost could then be used as a dynamic task priority similar to the deadline in EDF" [9].

In that regard, Henriksson et al. have designed a value-based dynamic scheduling of multiple MPCs where they used the "optimisation cost function as the value function" [47]. This work resembles the admission control work carried-out by Lu et al.'s in [77] in the sense of enforcing virtualised reward values for jobs to improve jobs admission and processing on time. However, it lacks featuring decision-making algorithms for acquiring additional computing processors if the local cluster is predicted to lack performance with respect to the RTS objectives. Besides, Henriksson et al.'s work was concerned with periodic and pre-defined jobs.

In the field of ERP, we can refer to Oracle's data sheet "Planning the scheduler" as a "non-stop process, driven by the arrival of new orders [jobs], unforeseen events, ... and changes in resource availability" [94]. The *changes* in the above quote can be related to the local cluster's over-utilisation, or to be precise, the prediction of our cluster's incapability to meet jobs deadlines given past and current jobs arrival and completion rate with respect to our RTS objectives. Oracle's data sheet in [94] also suggests that the planning process can be "proactive and reactive". Our prediction and optimisation algorithms are considered the proactive part of proposed algorithm, and the feedback loop, including the admission control decision, is the reactive part.

In chapter 6, we propose novel rent and purchase value-based admission control algorithms. The algorithms complement our proposed approach to address our industrial ERP requirement in acquiring additional computing processors while complying with financial constraints and incentives. Since our value-based algorithms include task dependencies within its scheduling, this can relate to actual industrial cases when dealing with "multi-step / complex orders that allows order time and sequence dependencies to be honoured in the schedule" [95]. Aiming for high completion ratio and minimum missed appointments are also advocated for in [95] which are reflected by our RTS objectives.

2.7 Summary

Traditional RTS approaches require complete knowledge of the scheduling parameters of the workload (task-sets) apriori in order to generate system schedules before jobs arrive [73]. Some work on earlier RTS policies included on-line admission control heuristic in order to enable an efficient scheduling performance such as the work in [96]. Although some works do not necessarily require the arrival times of jobs, but they still demand some RTS parameters e.g. resource requirements and task dependencies etc. apriori. Alternatively, we saw that some related works assume only periodic jobs and/or a pre-emptive RTS environment.

Traditional algorithms are considered OL [105, 73] because there is no automatic mechanism in updating the scheduler with new events in the system e.g. arrival of jobs with unknown computation time values and task dependencies. Such RTS policies can depend on the manual intervention of system administrators when QoS levels are violated or in the event of system overload which can lead to violating QoS levels. Hence, the motivation for our proposed algorithm has emerged in enabling the automation of admission control throughout unexpected events. According to our literature survey and to the author's best knowledge, there were no RTS solutions that have directly addressed handling task dependencies and real-time ERP applications with respect to our RTS objectives.

Chapter 3

Problem Formulation

3.1 Basic Definitions

In order to better introduce our thesis hypothesis and research questions in the coming sections, it is beneficial to introduce our basic definitions which will be used later on in this chapter and the rest of the thesis.

3.1.1 Open-Loop Admission Control

An algorithm that practises admission control without knowledge of the system's performance with respect to RTS objectives.

3.1.2 Feedback Admission Control

An algorithm that practises admission control with some knowledge of the system's performance with respect to RTS objectives.

3.1.3 HPC Cluster Processor Utilisation

The ratio of the number of busy processors to the total number of processors per time unit, see Equation 3.1.

$$PU = \frac{\#BusyProcessors}{ClusterSize} \tag{3.1}$$

3.1.4 Slack Values

We have already defined in chapter 1 that a job can contain a number of dependent tasks. Slack values measure how early or late a job/task is since its arrival to the system until it completes processing from the cluster.

For task slack values, tS, see Equation 3.2:

$$tS = tD - Time - tC' \tag{3.2}$$

tD is the task's deadline value, $Time$ is the real time clock (cycle start) and tC' is the task's remaining computation time.

For job slack values, jS, see Equation 3.3:

$$jS = jD - Time - jC' \tag{3.3}$$

jD is the job's deadline value where jC' is its remaining computation time.

3.1.5 RTS Objectives

- Manage the trade-offs, in real-time, between the maximisation of HPC cluster processor utilisation and slack values.
- Further objectives will be included in chapter 5 and chapter 6 once we address the issues of task dependencies and extending the HPC cluster via renting and buying additional processors, respectively.

3.2 Research Hypothesis and Questions

The thesis hypothesis is:
Given an open-loop admission control baseline, can a feedback admission control algorithm improve the real-time performance in increasing the HPC cluster processor utilisation and slack values of jobs with task dependencies with abilities in extending the HPC cluster via rent and buy algorithms.
We can outline the four research questions this thesis will address in the prescribed order leading towards answering the aforementioned hypothesis.

1. Can we construct a dynamic mathematical model which captures the (non-linear) behaviour between task-sets computation time values (system input) and their slack values & processor utilisation (system outputs). If so, how can we validate its accuracy in order for it to be implemented as part of the proposed algorithm?
2. Once we validate our system model, can we design an intelligent admission control algorithm for managing the trade-offs between the two RTS objectives without prior knowledge of jobs RTS parameters?
3. Can the proposed algorithm deal with task dependencies (inside jobs) as a scheduling constraint and the emerging RTS objectives such as reducing wasted processor utilisation (WPU)[1]? Can the proposed algorithm manage the trade-offs between the RTS objectives without prior knowledge to any RTS parameters? If so, can the proposed algorithm outperform

[1]We designed WPU as an RTS metric when addressing the issue of task dependencies where chapter 5 looks into it in more details.

baseline admission control algorithms such as random-based and jobs-completion-ratio-based etc.?

4. Can the algorithm support real-time rent and buy decisions for acquiring additional computing processors to support the local cluster in managing the trade-offs between the RTS objectives and increase the overall $SystemCredit$[1]?

3.3 Thesis Structure

We have introduced our research scope in chapter 1. In chapter 2, we reviewed the literature with respect to related works on intelligent admission control scheduling. This chapter includes the research hypothesis, questions and structure. We shall start with our first research question in chapter 4 because it represents the modelling foundation for the rest of our work carried-out in this thesis. We will also address the second research question in chapter 4 where we review and discuss our proposed algorithm's design and implementation as a general method for RTS systems resembling our industrial WMS.

In chapter 5, we will address task dependencies as a scheduling constraint. We will evaluate how our proposed algorithm can offer some answers to the RTS issues raised in this realm. We will also show how we extended our proposed algorithm in chapter 4 to adapt towards task dependencies as a scheduling constraint and handle jobs QoS levels as well. This covers our third research question.

In chapter 6, we will show how we extended our novel admission control algorithm, in chapter 5, to include ERP applications. This is to support our industrial organisation in making rent and buy decisions for acquiring additional computing processors when the local HPC cluster lacks, or is predicted to lack, performance in managing the trade-offs between the RTS objectives. We will evaluate our proposed algorithm and we will compare renting against buying admission algorithms with respect to the RTS objectives and the financial consequences as well. This covers our fourth and final research question leading to address the thesis hypothesis.

Conclusions & further works for our work in this thesis are found in chapter 7.

[1]We designed $SystemCredit$ as an RTS metric when addressing ERP optimisation problems where chapter 6 looks into it in more details.

Chapter 4

A Novel Feedback Admission Control Scheduling Algorithm

The purpose of this chapter is to discuss the feedback admission control (FAC) algorithm which is our adopted approach in this thesis. This algorithm is not only tailored for our industrial system, but is considered a general approach for all WMSs that have similar RTS objectives and requirements as outlined in chapter 1 and chapter 2. Our algorithm is based on feedback control, see Figure 4.1, which was motivated for in our literature survey in chapter 2 as a promising approach for our industrial system and other similar RTS systems.

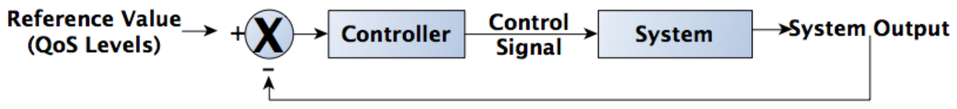

Figure 4.1: A block diagram of a typical feedback control system.

As any typical feedback control system, our admission control approach is based on two main components as depicted from Figure 4.1 which are the *controller* and the *system*. These two components will be briefly outlined here to provide a general picture of our approach to the reader. Afterwards, in-depth analysis and review of our FAC algorithm's components and design steps will be provided in the next sections.

The **controller** component is the admission controller of our FAC algorithm. It computes a control signal after comparing the monitored system outputs e.g. slack values and processor utilisation with their associated QoS levels, refer to section 4.2 for more details. The control signal represents the predicted processing capacity the HPC cluster can afford to provide without violating the QoS levels. The processing capacity is considered as the "optimal" computation time value which the admission controller uses for admission decision-making for the arriving jobs.

As we do not know jobs' RTS parameters apriori we cannot afford to control job release rates. We designed our control signal to be concerned with the computation time values as they can be known once jobs arrive so the admission controller can compute an instant admission control decision. If the arriving job has a computation time value larger than the optimal one then it is rejected. It is rejected because the admission controller predicted the arriving job will not meet its deadline and will cause other jobs to be delayed potentially causing deadline misses (undesired).

As for the other component, to control any **system**, any controller requires a linear mathematical model of the system under consideration [92]. We will refer to such a model as the **system model** for the rest of this thesis. For modelling computing systems, we have queueing theory which has been reviewed in chapter 2 as an impractical solution in admission control [100]. System identification constructs linear mathematical models approximating the behaviour between the system parameters [92, 46]. The admission controller uses such a model to capture any sudden events in the system e.g. jobs arrivals and cluster overload etc. so it can prevent QoS violations via admission decisions.

4.1 Design Steps

Figure 4.2-(a) summarises the design steps adopted in our FAC algorithm and they also make the design foundation for all feedback control RTS approaches. Each one of the design steps will be reviewed in details with respect to our algorithm which can be represented by Figure 4.2-(b). The FAC block, in Figure 4.2-(b), is a software layer which can be added to the WMS software algorithm. In fact, if we remove the FAC software layer, we will end-up with the OL representation of our industrial system, see Figure 4.3. OL here refers to the absence of feedback control mechanism which closes the loop as noticed in Figure 4.2-(b). We developed our own simulator for this work in Simulink environment in Matlab [86].

4.1.1 Task Model

We start with exploring the performance of our FAC algorithm with a simple task-set. This task-set assumes independent tasks, therefore the notion of jobs does not yet exist in our workload. Once we introduce task dependencies, in the next chapter, the notion of jobs emerges. Although we are primarily concerned with aperiodic task-sets, we are enforcing the assumption of periodic tasks at this early stage of the thesis. The reason for this is to evaluate the

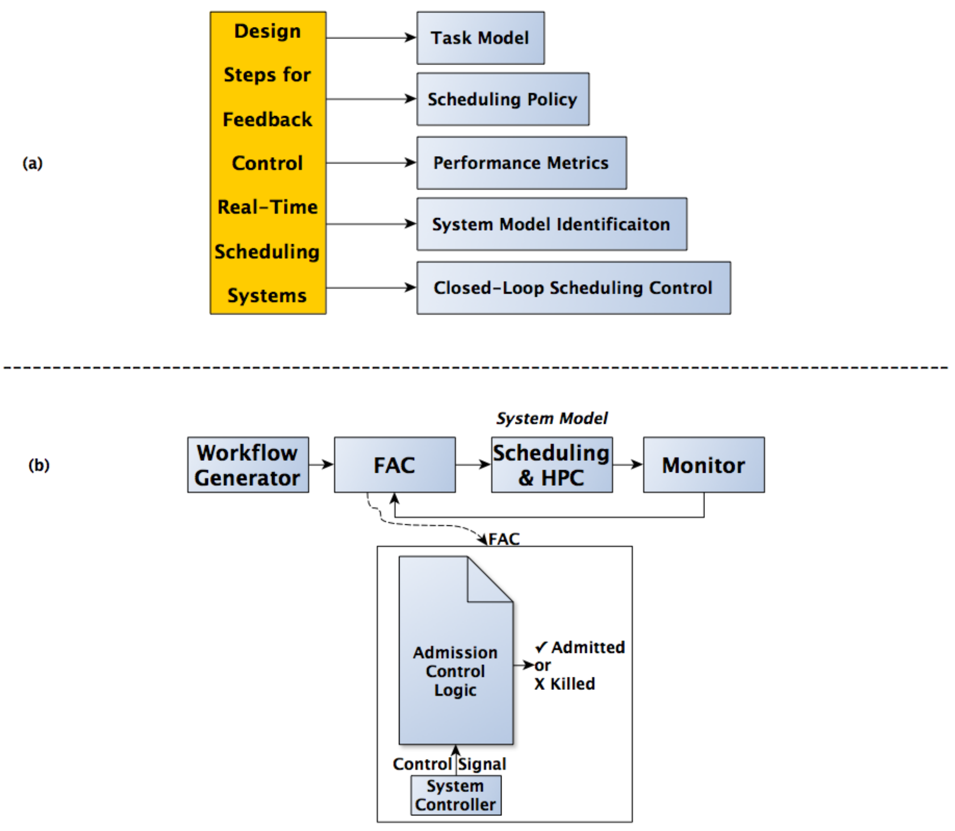

Figure 4.2: (a) Design steps for feedback control real-time scheduling systems. (b) A block diagram of our feedback admission control algorithm.

efficiency of our algorithm when using basic and advanced control laws at this stage before adding further constraints to our RTS problem.

A workflow task from our industrial system can be represented in a tuple like [tC_i, tD_i, tT_i] [8]. Where tC_i is the i^{th} task's computation time value, tD_i is the deadline and tT_i is the release rate. We borrowed two types of tasks from our industrial system which are Data and Design. Such types are similar in RTS parameters and are represented by Table 4.1. Please note that our algorithm is not only concerned with such task-sets, however it is an early exploration phase of it and we will see in the next chapter how it performs with different task-sets which can be more related to other industrial RTS task-sets.

The tD for all tasks is set to be the tC value multiplied by an arbitrary value of 1.2 chosen for this work. In other words, the tD of individual tasks is equal to their individual tC value plus 20%. We use the notion of task slack values, tS, as defined in chapter 3, see Equation 3.2. If a task finishes earlier than its deadline then a task has a positive tS value. A task will have a zero tS value if it finishes on time and if it is late then it will suffer from negative tS which is

Workflow Task Type	***tC*** **(hours)**	***tD*** **(hours)**	***tT*** **(hours)**
Data	[4-6], [12-15]	tC + 20%	4.0
Design	[4-6]	tC + 20%	0.8

Table 4.1: Task model of generated tasks [42, 43]. For Data tasks, a Round-Robin selection criteria is implemented inside each range, and random switching criteria is implemented between the two ranges. This can be noticed in Figure 4.4 (top). For Design tasks, they have only one range and values are selected via a random selection criteria.

undesired. We take the normalised value of tS values, ntS, for our results, see Equation 4.1.

$$ntS_i = \frac{tD_i - Time - tC_i'}{tD_i} \tag{4.1}$$

tC_i' is the i^{th} task's remaining computation time. The reasons for normalising tS are to show (1) what task has missed its deadline and (2) how far is it from its deadline; this can be positive, zero or negative values. Given Equation 4.1, ntS_i ranges between $-\infty$ and 1, where we want to maximise ntS_i to achieve lesser deadline misses.

If we apply the RTS schedulability bound equation of Liu and Layland in [69], see Equation 4.2, for our task-sets in Table 4.1, the processor utilisation, U, required is greater than 1, where C_i is task computation time values, and T_i is task release rates. This will inevitably cause processor over-utilisation and deadline misses.

$$U = \sum_{n}^{i} \frac{C_i}{T_i} \leq 1 \tag{4.2}$$

In other words, the scheduling of our workflow task-sets i.e. meeting all individual tasks deadlines is infeasible according to the schedulability equation test because in our case tC_i is always larger than tT_i. Although this test is for single-processor RTS systems only, we will see in later sections in this chapter how our industrial tasks will still miss their deadlines and suffer negative ntS values with many-processor clusters. We will notice in our forthcoming evaluations that our industrial system experiences cluster overload situations which is considered an indication that sometimes the number of HPC processors are not sufficient to process all incoming workloads leading to reduced ntS values which potentially causes deadline misses in the system.

4.1.2 Scheduling Policy

We have experimented with FCFS and EDF separately to observe the performance and benefits each policy brings with respect to our RTS objectives.

4.1.3 Performance Metrics

- Processor utilisation, (PU), see Equation 3.1.
- Normalised task slack values, (ntS), see Equation 4.1.

4.1.4 System Model and Identification

Classical control systems such as mechanical and electrical systems may have readily available differential equations as system models, unlike real-time computing systems [73]. A system model represents the OL relationship between system parameters (inputs and outputs). System parameters are chosen to, for instance, measure how tasks tC values (system inputs) affect the PU and ntS values (system outputs) of tasks that have completed processing. In typical mechanical and electrical systems, the relationships between the system parameters are usually governed by physical laws e.g. Newton's law. However, this is not the case with computing systems parameters [39].

If we look at Figure 4.3 which is the OL version of our industrial system, the system inputs stem from the Workflow Generator block and the system outputs are monitored via the Monitor block. We can derive linear mathematical models using system identification regression equations [71] that can capture the (non-linear) relationship between the system parameters. Such models are considered as the foundation for controller design [92], and will replace the scheduling and processing block as shown in Figure 4.2-(b) and later on in this thesis with a System Model block.

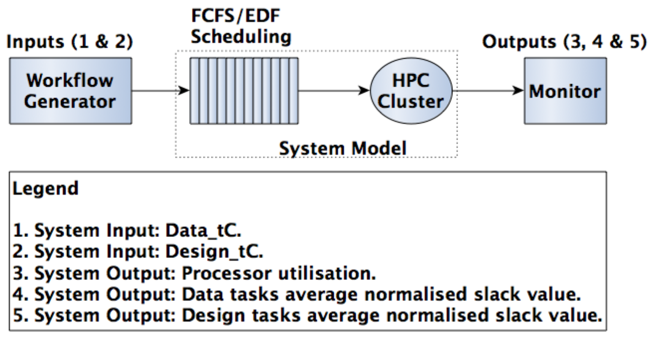
FCFS/EDF Scheduling
Inputs (1 & 2)
Outputs (3, 4 & 5)
Workflow Generator
HPC Cluster
Monitor
System Model
Legend
1. System Input: Data_tC.
2. System Input: Design_tC.
3. System Output: Processor utilisation.
4. System Output: Data tasks average normalised slack value.
5. System Output: Design tasks average normalised slack value.

Figure 4.3: A block diagram of the open-loop (OL) version of our industrial system with emphasis on the System Model components. The System Model block represents the system identification model of the scheduling and processing dynamics of our industrial system which the FAC block uses in its admission control decision-making process in Figure 4.2-(b).

In general system identification theory [85], we consider a non-linear system like our industrial WMS with an output, $y(t)$, that follows a prescribed trajectory in response to a known input, $u(t)$. The output can be a vector of the monitored *ntS* values and *PU* and the input can be the incoming tasks *tC* values. The system dynamics, then, can be represented by Equation 4.3:

$$\frac{dx(t)}{dt} = f(\mathbf{x}, \mathbf{u}); \mathbf{y} = g(\mathbf{x}, \mathbf{u}); \tag{4.3}$$

Both x and y represent the system's input and output states respectively. Both functions f and g, which can be non-linear, are mathematical representations of the system dynamics via measurements. If the system is at an equilibrium condition, then a small perturbation to the input, Δu, which is the control signal, leads to a small perturbation in the output, Δy, see Equation 4.4 [85]:

$$\Delta\dot{\mathbf{x}} = \frac{df}{dx}\Delta\mathbf{x} + \frac{df}{du}\Delta\mathbf{u}; \Delta\mathbf{y} = \frac{dg}{dx}\Delta\mathbf{x} + \frac{dg}{du}\Delta\mathbf{u}; \tag{4.4}$$

In other words, this small perturbation to the input can be a correcting signal in order to steer the system's dynamics towards meeting the QoS levels. We have implemented a widely known system identification technique called non-linear autoregressive exogenous (NARX) [71] for capturing the impact of arrived tasks *tC* values on scheduling and processing. We implemented NARX correlation functions in our simulated system inputs and outputs because, as any other computing system, such relationship is non-linear and NARX is able to capture such non-linearity in its modelling technique, more details can be found in [71].

The system parameters (inputs and outputs) were simulated for 200 hours for the modelling purposes to represent around two weeks worth of our industrial WMS's RTS dynamics. We selected the system inputs to be the *tC* values, depicted from Table 4.1, for both types of tasks (Data and Design), see Figure 4.4, in order to study their effects on our RTS objectives by monitoring system outputs in Figure 4.5. We simulated *tC* in the OL simulation model, see Figure 4.3, in order to capture correlations (system dynamics) between *tC* (inputs) and *PU* & *ntS* (outputs).

For general RTS systems, any parameters can be selected and simulated for a particular period in order to model as much RTS dynamics as possible so the admission controller can capture such dynamics via the obtained system model. The simulated inputs and outputs are represented in Figure 4.4 and Figure 4.5, respectively. Using the OL simulation model for capturing the system dynamics is one of the three system identification approaches outlined in section 2.2.

The inputs were simulated on a 10-processor cluster to capture the clus-

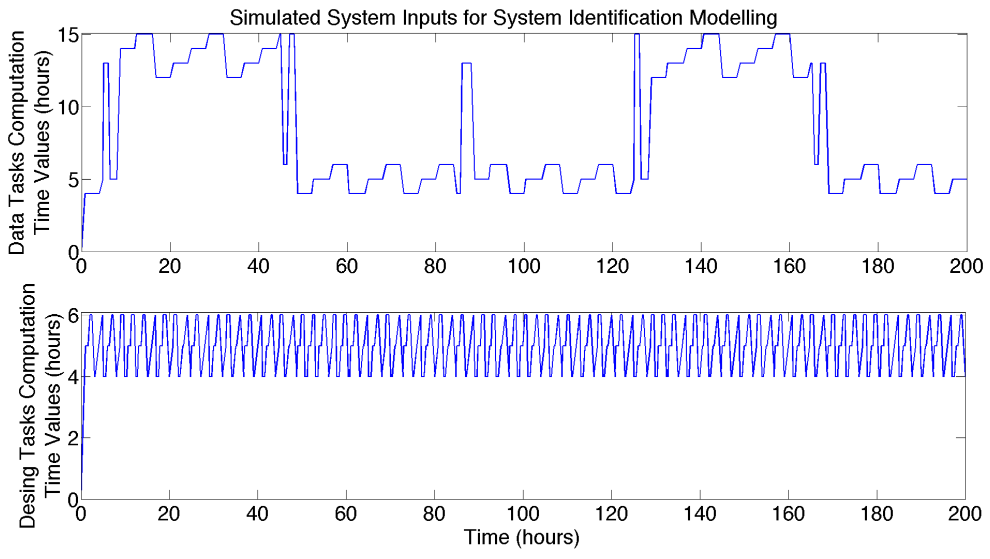

Figure 4.4: Simulated system inputs for system identification modelling. We can notice for the Data tasks (top plot) the Round-Robin selection criteria within each range, and the random switching mechanism between the two ranges.

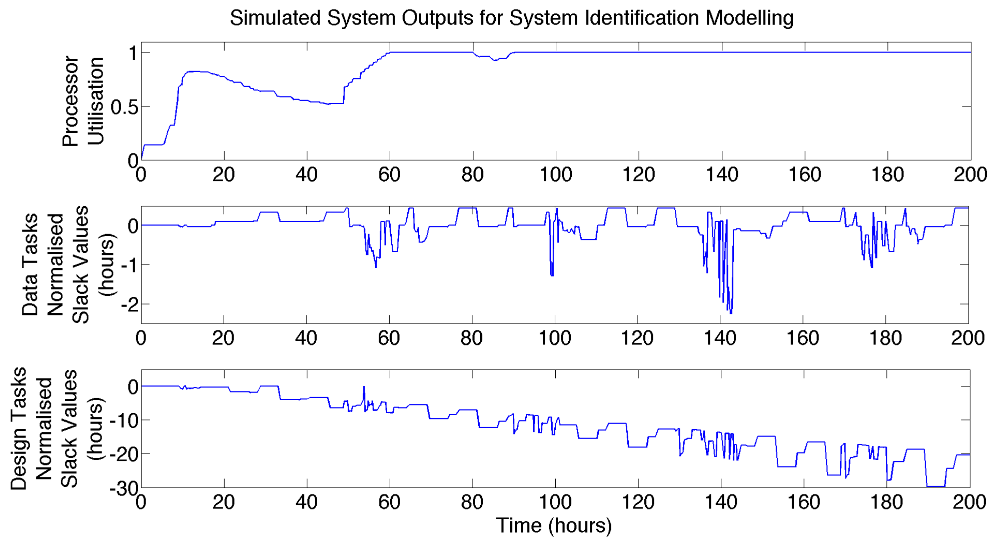

Figure 4.5: Simulated system outputs for system identification modelling.

ter overload situations. Thus, the model obtained will be restricted to the dynamics captured in the used cluster size. We notice from Figure 4.5 that Design tasks suffer worse ntS values than their Design peers, and this is witnessed in our industrial system actually. An argument can be raised here regarding the identification model we did for high workloads when we simulated the system inputs and outputs on a small cluster size. We did not simulate data of higher cluster sizes to capture the cluster under-utilisation situations. Modelling different workloads can be marked as a future work for this chapter.

System identification has allowed us identifying the RTS dynamics in the form of a set of transfer (system) functions in terms of complex frequency-domain representation, z-domain, of linear time-invariant systems. In control theory, it is preferred to use transfer functions in the z-domain representation, not the time-domain, because a controller is concerned with the varying dynamics of the system's parameters regardless of the notion of time. We added the transfer functions as the System Model block in Figure 4.2-(b). The transfer function from Data tasks tC values (input 1) to PU (output 1) is:

$$\frac{-1.78e^{-15}z^6 - 1.24e^{-3}z^5 + .01z^4 - 0.03z^3 + 0.02z^2 - 1.33e^{-3}z - 1.70e^{-3}}{z^7 - 4.14z^6 + 7.00z^5 - 6.17z^4 + 2.99z^3 - 0.75z^2 + 0.08z} \tag{4.5}$$

The transfer function from Data tasks tC values (input 1) to ntS_{Data} values (output 2) is:

$$\frac{1.77e^{-15}z^6 + 3e^{-5}z^5 - 6.00e^{-5}z^4 + 4.63e^{-5}z^3 - 1.98e^{-5}z^2 + 6.25e^{-6}z - 1.18e^{6}}{z^7 - 4.14z^6 + 7.00z^5 - 6.17z^4 + 2.99z^3 - 0.75z^2 + 0.076z} \tag{4.6}$$

The transfer function from Data tasks tC values (input 1) to ntS_{Design} values (output 3) is:

$$\frac{2.66e^{-15}z^6 - 9.38e^{-4}z^5 + 4.01e^{-4}z^4 - 6.74e^{-3}z^3 + 5.58e^{-3}z^2 - 2.27e^{-3}z + 3.68e^{-4}}{z^7 - 4.14z^6 + 7.00z^5 - 6.17z^4 + 2.99z^3 - 0.75z^2 + 0.08z} \tag{4.7}$$

The transfer function from Design tasks tC values (input 2) to PU (output 1) is:

$$\frac{-8.88e^{-16}z^6 + 0.38z^5 - 1.14z^4 + 1.30z^3 - 0.71z^2 + 0.20z - 0.03}{z^7 - 4.14z^6 + 7.00z^5 - 6.17z^4 + 2.90z^3 - 0.75z^2 + 0.08z} \tag{4.8}$$

The transfer function from Design tasks tC values (input 2) to ntS_{Data} values (output 2) is:

$$\frac{5.13e^{-4}z^5 - 1.66e^{-3}z^4 + 2.12e^{-3}z^3 - 1.33e^{-3}z^2 + 4.09e^{-4}z - 5.01e^{-5}}{z^7 - 4.14z^6 + 7.00z^5 - 6.17z^4 + 2.99z^3 - 0.75z^2 + 0.077z} \tag{4.9}$$

The transfer function from Design tasks tC values (input 2) to ntS_{Design} values (output 3) is:

$$\frac{-0.01z^5 + 0.03z^4 - 0.04z^3 + 0.03z^2 - 0.01z + 1.89e^{-3}}{z^7 - 4.14z^6 + 7.00z^5 - 6.17z^4 + 2.99z^3 - 0.75z^2 + 0.08z} \tag{4.10}$$

It is worthy to mention that lower order models were investigated in this work. However, the current model order has resulted in a closer match to the industrial system's simulated parameters of 2 inputs and 3 outputs where most systems in the literature varied from single to double input/output. Also, the performance of a higher-order identification model advances as the number of parameters in the identification problem can yield to infinity [31]. An argument can be made here regarding the time-variant nature of our industrial WMS, literature of control theory supports linear time-invariant difference equations [46, 71]. We used Matlab's System Identification toolbox [86] to convert the simulated data into a linear mathematical representation (preferably frequency-domain) that is compatible with control theory. More information can be found in [46].

In order to start using the obtained transfer functions as the system model in our FAC algorithm, it needs to be validated [92]. Such validation is achieved if the system model passes an accuracy test [46]. This accuracy reflects how accurate the system model is in representing the system's RTS dynamics. If we simulate the system dynamics for 200 hours, then in system identification we take one part of this data for constructing the system model itself. The other part is used for validating the accuracy of the constructed model. The accuracy, refer to Equation 4.11, of our system model has been tested via a process called Model Validation [46, 92].

$$R^2 = 1 - \frac{var(y - \widehat{y})}{var(y)} \tag{4.11}$$

The variance of system outputs, $y(k)$, is $var(y)$, and $\widehat{y}$ is the estimated value of the system outputs. R^2 ranges from 0 to 1; zero means the model does not work, any better, than just taking the mean value of y to estimate $y(k)$. In general identification theory, we look for system models with $R^2 \geq 0.8$ [46]. The reason it is not 100% because it is difficult to fully capture the non-linear behaviour of computing systems and 80% model accuracy is, in general, considered an acceptable threshold [46]. Our identified model has an accuracy of 82.71% which passes the accuracy threshold for it to be adopted in our FAC algorithm. We can argue at this stage that the more accurate the system model is i.e. nearer to 100% accuracy, the better the admission control performance. However, studying the non-linearities in system identification models is a research field by itself and our residual analysis indicates the system model fits within the 90-95% confidence interval, see Figure 4.6.

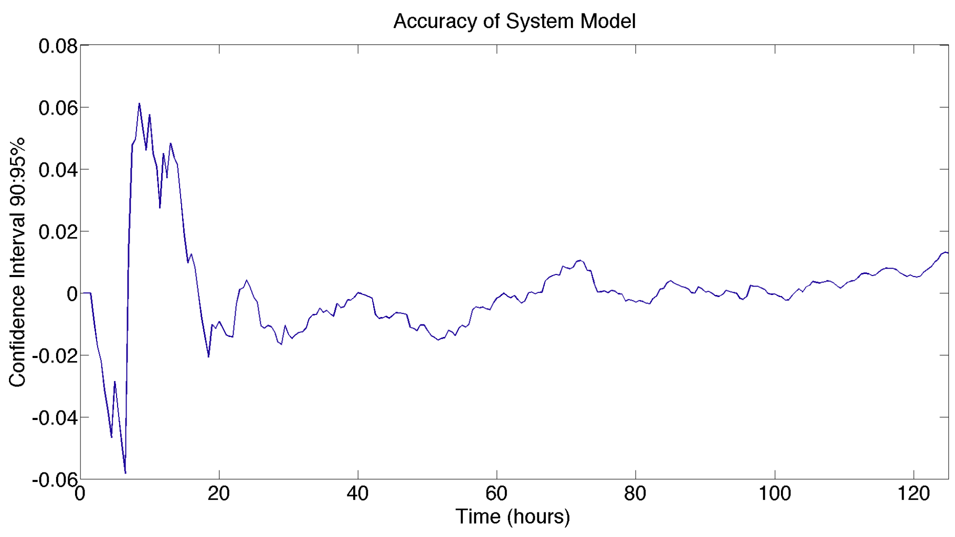

Figure 4.6: Our system identification model accuracy: measured minus simulated data. The reason the x-axis is not the full 200 hours is because in system identification we allocate one part of the measured data for the model estimation and another for model's accuracy validation [46].

Once we obtain a model and validate its accuracy, we can start with our controller design and decide upon the control actions (signals) [57]. The next subsection will look into this in more details. At this stage we can address our first research question which is:
"*Can we construct a dynamic mathematical model which captures the (non-linear) behaviour between task-sets computation time values (system input) and their slack values & processor utilisation (system outputs). If so, how can we validate its accuracy in order for it to be implemented as part of the proposed algorithm?*"
We can answer it with *yes*.

4.1.5 Closed-Loop Scheduling Control

This step starts from the identification model of the OL system. Controller design i.e. the design of the control signal varies between two popular areas as far as FAC applications are concerned with:

- Basic controller design such as PID and its combinations e.g. P and PI etc. [43, 74, 75].
- Advanced controller design such as MPC and LQR [28, 42, 78].

We can notice from chapter 2 that both PID- and MPC-based FAC algorithms are two popular approaches in the literature. For instance, Lu et al. have started their PID-based FAC work with a single processor and a single-objective optimisation (minimise deadline miss ratio) in [75]. They adopted a multiple

PID approach once they shifted to multi-objective optimisation in [76] where they implemented two PID controllers one for each RTS objective (maximise processor utilisation and minimise deadline miss ratio). However, later on when multi-processor scenarios were examined by Lu et al., MPC has offered promising results for the same RTS multi-objective optimisation problem [78].

This very work of Lu et al. has inspired us adopting MPC supporting admission control decisions for RTS [42] as MPC-based admission control was suggested in [78] where they suggested using MPC for FAC purposes. We developed our own novel MPC-based FAC algorithm for admission control decision-making where our scientific contributions in this chapter are:

- Implementing a classical modelling approach known as system identification for modelling the RTS dynamics of our industrial WMS which is a pre-requisite for our admission controller design.

- Introducing the novel idea of using optimal task computation time values, $tC_{Control}$, as the control signal for admission control. Other related works used metrics such as tasks release rate and assumed knowledge of task-sets apriori, see chapter 2 for more details. The benefit of using $tC_{Cotnrol}$ is to allow the admission controller to compute admission decisions without requiring any information about tasks RTS parameters apriori. Instead, the admission controller can compare the tC value of the newly arrived tasks with $tC_{Control}$ where the latter is mapped in real-time with respect to system QoS levels.

- Designing our MPC-based FAC algorithm, [42], is a novel improvement on Lu et al.'s work on multi-objective RTS optimisation. Lu et al. addressed a similar RTS problem (multi-processor and multi-objective) to ours but the difference of their approach is that they assume some knowledge of RTS parameters apriori and they also control task release rates. We have also implemented our own version of PID-based FAC algorithm using our novel notion of $tC_{Control}$ as a control signal and have evaluated the algorithm against its MPC peer. The motivation for these evaluations is to test basic (PID) and advanced (MPC) FAC algorithms for meeting our RTS objectives.

The next two sections will review the controller design aspect of both of our novel PID- and MPC-based FAC algorithms with respect to our RTS objectives and then will be followed by the evaluations in section 4.4.

4.2 Design and Implementation of PID-based Feedback Admission Control

A PID controller is composed of Proportional, Integral and Derivative parts, see Figure 4.7-(a). The main objective of such a controller is to minimise the

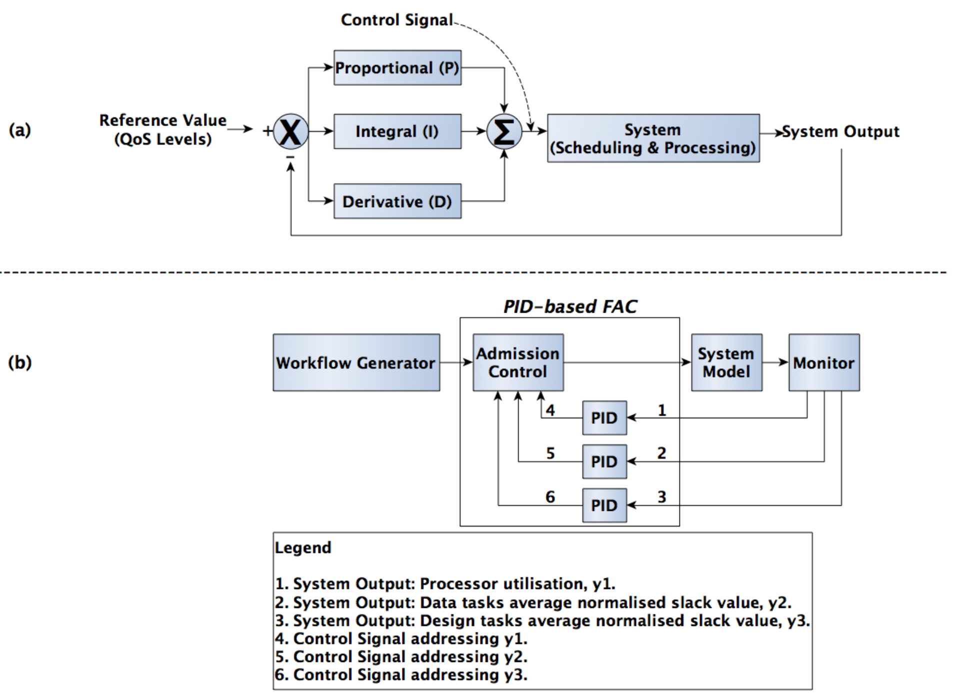
(a)
Control Signal
Proportional (P)
Reference Value (QoS Levels)
Integral (I)
Σ
System (Scheduling & Processing)
System Output
Derivative (D)
(b)
PID-based FAC
Workflow Generator
Admission Control
System Model
Monitor
PID
Legend
1. System Output: Processor utilisation, y1.
2. System Output: Data tasks average normalised slack value, y2.
3. System Output: Design tasks average normalised slack value, y3.
4. Control Signal addressing y1.
5. Control Signal addressing y2.
6. Control Signal addressing y3.

Figure 4.7: (a) A schematic representation of PID control. (b) Our novel PID-based feedback admission control algorithm where the System Model block is the System Identification model obtained from subsection 4.1.4.

system error in order to achieve the desired QoS levels expressed as a function of time. In other words, it takes into consideration the current system error (the proportional part) along with the past error (the derivative part) and the rate of change (the integral part) [92]. The desired QoS levels in our industrial system, and in general industrial RTS systems as well, are:

- PU = 100%.
- ntS_{Data} = -1. The reason for the small, negative QoS level is because if we have a zero or positive QoS level, the controller will become aggressive. Aggressive admission control actions include rejecting a high number of tasks in order to strictly avoid any deadline misses (negative slack values). As we are dealing with soft real-time tasks in a system which is expected to experience overload situations. We allocate a small, negative QoS level to avoid aggressive admission control actions which can potentially lead to processor under-utilisation (undesired) [76].

- ntS_{Design} = -1.

Our novelty in the implementation of PID control is in utilising $tC_{Control}$ as part of the controller design. In this case, the PID controller uses its three correcting parts where their sums constitute the $tC_{Control}$ correction as a function of time. See Equation 4.12.

$$u(t) = K_p e(t) + K_i \int_0^t e(t).dt + K_d \frac{d}{dt} e(t) \tag{4.12}$$

We can analyse the PID control equation in terms of our industrial system:

- $u(t)$: is the control signal e.g. the new, optimal $tC_{Control}$ value which is used in order to reach the desired QoS levels via admission control. In Figure 4.7-(b), the third PID controller (which receives Signal 3 and computes Signal 6) can, for instance, compute a control signal with a value of 60. This value means that the system can currently admit Design task(s) with up to the value of 60 tC units without violating its QoS level.

- K_p: is the proportional gain, its value can be tuned.

- $e(t)$: is the system error, which is the difference between the current system output value and the desired one i.e. QoS level. Each PID controller computes the system error for the output it is associated with. For instance, a system error of -10 computed by the third PID controller means that the system's ntS error for Design tasks is 10. This error refers to the fact that such tasks are 10 times later in average than they should be in comparison to the QoS level of their type.

- K_i: is the integral gain, its value can be tuned.

- $\int_0^t e(t).dt$: is the sum of the system errors over a certain period of time, this is to feed the controller the rate of change i.e. how the system error evolves over time.

- K_d: is the derivative gain, its value can be tuned.

- $\frac{d}{dt}e(t)$: is the derivative of the system error over time, in other words the slope of error over time.

Different systems require certain features that eliminate the need of incorporating the full PID design. This is to reduce measurements overhead. The time granularity of our industrial system is in minutes and it follows that implementing the full PID control will not cause any issues in having the admission decision being delayed by micro-seconds computed by the admission controller. More analysis of PID controls can be found in [5]. That is why we will notice very similar performance across the P, PI and PID controllers in the evaluations, section 4.4.

Since our PID-based FAC algorithm is inspired by Lu et al.'s approach in [75] which addresses a similar RTS problem as ours, we adopted their PID controller configurations, see Table 4.2. We can argue here that exploring the science of controller tuning [89, 92] can yield to improved controller performance. However, we decided to use Lu et al.'s PID parameter values because of the similarity in RTS problem-solving. We have also implemented P- and PI-based FAC algorithms to evaluate against the full PID-based algorithm. In the P-based algorithm, we made the I and D control parts equal to zero. Consequently in the PI-based algorithm, we made the D control part equal to zero. Making any part to zero implies *switching it off*. In control theory, as

PID Parameter	**Parameter Tuning Value**
Kp	0.5
Ki	0.05
Kd	0.1

Table 4.2: PID parameters tuning as depicted from Lu et al. in [75].

far as PID controller stability is concerned, we explore the controlled system's transfer function's poles. The transfer function provides a basis for determining important system response characteristics [115]. Our controller has followed Lu et al.'s PID stability criterion which are described in details in [76].

The Admission Controller block receives Signals 4, 5 and 6 from the PID blocks as depicted from Figure 4.7-(b). Signal 4 is computed to maximise PU. At the case of receiving a Data type task, the Admission Control block maps Signals 4 and 5 to an admission decision. When a Design task arrives, the Admission Control block maps Signals 4 and 6 for admission. The reason for having Signal 4 part of the mapping for both types of tasks is to address the first component of our QoS levels, PU. Since each control signal computed by the PID blocks is independent from one other, it was necessary to map them, accordingly, in every admission decision.

The novel PID-based FAC logic is:

```
if arrived task is Data
    if (task computation time value <= control signal 5) &&
        ((task computation time value * control signal 4) <= 1)
        admit task;
    else
        kill task;
    end
elseif arrived task is Design
    if (task computation time value <= control signal 6) &&
        ((task computation time value * control signal 4) <= 1)
        admit task;
    else
        kill task;
    end
end
```

We can see there are two admission criterion for each task to be admitted into the system for scheduling and processing. First criteria detects whether *tC* of newly arrived task(s) exceeds the mapped signal, $tC_{Control}$, of the appropriate task type (Data or Design). If so, then it is believed that the newly arrived task will decrease the system's overall *ntS* value of that particular type beyond their QoS level. Thus, the task is rejected at this stage before going through the second admission criteria. This admission criteria is motivated for our industrial system because it does not require the knowledge of any of the tasks RTS parameters apriori.

Second, the *PU* control signal (Signal 4) increases the likelihood of accepting a task simply in order to reach the desired QoS level of 100%. However, in the case of processor over-utilisation, a task can be rejected due to Signal 4's over-utilisation being mapped to the Admission Control block. The PID block for *PU* computes the available processing capacity of the cluster via real-time monitoring. Then, it maps its control signal for tasks admission by multiplying it with the total processing time units currently being processed in the cluster. This way, the newly arrived task's *tC* value is compared to control signal 4 where if they produce $> 100\%$ *PU* then the task is rejected. It is rejected on the basis that this task will cause over-utilisation which is an undesired RTS result causing deadline misses (decreased *ntS* values).

This way, enforcing these two criterion in parallel assures that a task is accepted without compromising our QoS levels [43]. Handling rejected tasks is not present in this work. Currently in the industrial WMS and as mentioned earlier in chapter 2, when a particular task causes the scheduling performance to degrade, the system administrator manually *kills* the whole job that contains this task. This is a serious flaw in the industrial system. However, it is part of our research in the future to design a mechanism that allows re-admission of rejected tasks in an optimised manner with respect to the RTS objectives.

4.3 Design and Implementation of MPC-based Feedback Admission Control

In control-theoretic terms, our industrial system is a non-linear MIMO system with a multi-objective RTS optimisation problem. We require a control approach that generalises directly to such system structures. MPC resembles the PID algorithm in Figure 4.7-(a) but replacing all of the PID blocks with a single MPC block, see Figure 4.8-(a).

An advantage for MPC type of control is its ability to adhere to hard constraints on the control signal, $tC_{Control}$. For instance, we designed a constraint where $tC_{Control} \leq 0$ because it is meaningless to use a negative computation time value. This flexibility in MPC design makes it a better candidate for our admission control work than other advanced control algorithms such as LQR. LQR lacks enforcing such explicit constraints.

MPC has two main parts: prediction and optimisation. Our novelty in the implementation of MPC control is in utilising $tC_{Control}$ as part of the controller design. After conducting prediction and optimisation, the MPC block computes its $tC_{Control}$ and feeds it to the Admission Control block in order for the latter to apply it for admission purposes.

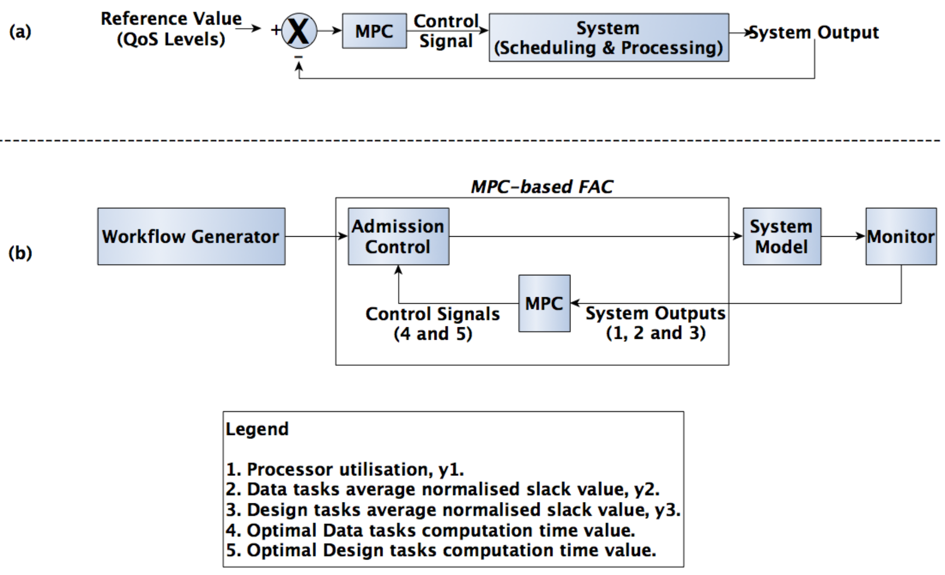
(a)
Reference Value (QoS Levels)
+
X
-
MPC
Control Signal
System (Scheduling & Processing)
System Output
(b)
MPC-based FAC
Workflow Generator
Admission Control
System Model
Monitor
MPC
Control Signals (4 and 5)
System Outputs (1, 2 and 3)
Legend
1. Processor utilisation, y1.
2. Data tasks average normalised slack value, y2.
3. Design tasks average normalised slack value, y3.
4. Optimal Data tasks computation time value.
5. Optimal Design tasks computation time value.

Figure 4.8: (a) A schematic representation of MPC. (b) Our novel MPC-based feedback admission control algorithm where the System Model block is the System Identification model obtained from subsection 4.1.4..

Each MPC control action, $tC_{Control}$, is computed at every sampling instant/-cycle. This is done by solving a finite horizon OL control problem using the current system state as the initial state. The controller predicts how much each system output, y_j, deviates from its QoS level, r_j, within the prediction horizon (sampling cycles), P. MPC multiplies each deviation by each system output's weight, w_j^y, and computes the weighted sum of squared deviations,

$S_y(k)$, refer to Equation 4.13.

$$S_y(k) = \sum_{i=1}^{P} \sum_{j=1}^{n_y} \Big[w_j^y [r_j(k+i) - y_j(k+i)] \Big]^2 \tag{4.13}$$

P Prediction horizon. It refers to the number of sampling cycles over which the controller computes its control signal with respect to the RTS objectives. The default value is 10 in MPC toolbox in [86]. In other words, the MPC predicts the system performance with respect to its RTS objectives for the next 10 time units and compute its control signal based on such a prediction.

n_y Number of system outputs = 3 [PU, ntS_{Data} and ntS_{Design}], see Figure 4.8-(b).

w_j^y Weight for system outputs y_j. This is to steer the controller's direction on favouring the QoS level of one parameter over another. Our Weights for the system outputs = [1, 0.5, 1] respectively; Data tasks have lesser weight than their Design peer. This is because according to our simulated outputs of the industrial system, Data tasks suffered lesser severity in ntS values than their Design peer, see Figure 4.5, which is actually the case currently in the industrial system.

r_j QoS levels for each system output. They are represented in a vector = [1, -1, -1] for PU, Data and Design ntS values, respectively. We have already explained and motivated the negative QoS levels for ntS values in section 4.2.

$r_j(k+1) - y_j(k+1)$ is the predicted deviation, system error, at future instant *k+1*; QoS level (minus) system output. The MPC computes our industrial system's system error by comparing the QoS levels vector with the three monitored outputs in real-time represented in a vector as well. For instance, a system error (deviation) vector of [20, 5, 10] means the system is 20% under-utilised and the ntS value difference for Data and Design tasks are 5 and 10, respectively. The ntS value errors refer to the fact that such tasks are X times later in average than they should be in comparison to their associated QoS levels.

k Current sample interval.

MPC applies its QP solver, [104], to each system output simultaneously. The sum iterations are equal to the number of system outputs, n_y. Each system output and its associated QoS level, r_j, and weight, w_j^y, are used in their particular iteration. More details on MPC controller design are found in the MPC toolbox documentation of [86]. The most commonly used MPC is linear

MPC where the dynamics are linear (or rather linearised) and a quadratic objective is used. A QP solver deals with the optimisation problem including a quadratic objective. A QP problem has an objective which is a quadratic function of (1) the control variable and (2) constraints which are all linear (linearised) functions of the control signal. This optimal control approach deals with the optimisation problem by minimising the quadratic cost function.

As far as the MPC stability is concerned, the controller went through a number of stability tests which are part of the MPC toolbox in Matlab. The MPC tests have three criterion: pass, warning (a pass with a suggestion) and error. The performed MPC tests were the following (an extensive description of each test can be found in the MPC toolbox documentation [86]):

- **MPC Object Creation**: it involves simple validity and consistency checks, such as signal dimensions and non-negativity of weights, w_j^y. Additional diagnostic checks occur during creation of the MPC controller, such as verification that the controller states are observable.
- **QP Hessian Matrix Validity**: it is concerned with the QP numerical accuracy. In order for the outputs to be controlled, they, individually, must respond to at least one manipulated variable (the control variable i.e. $tC_{Control}$) within P, the prediction horizon.
- **Controller Internal Stability**: the controller is internally stable if all its modes (predictive states) are exponentially stable or pure integrators (easily observable). The MPC can be fine-tuned to be internally stable via changing the P, w_j^y and sampling cycles, k. A controller may appear nominally stable but will perform poorly in practice if the System Model and $tC_{Control}$ create exponentially unstable modes.
- **Closed-Loop Nominal Stability**: the feedback communication between the plant (i.e. System Model) and controller should be stable for the nominal case i.e. when the controller's prediction model represents the plant perfectly and no constraints are active.
- **Closed-Loop Steady-State Gains**: this test is to determine if the controller forces all controlled outputs to their QoS levels at steady state, in the absence of constraints.
- **Hard Manipulated Variable Constraints**: the controller should always satisfy hard bounds on the manipulated variable ($tC_{Control}$) or its rate of change. We designed a hard bound for $tC_{Control}$ not to include negative values. Also, we chose not to specify any desired rate of change. For instance, there might be instances where the MPC predicts and computes $tC_{Control}$ to be 150 computation time units and then after

some time and due to, say, sudden arrival of jobs the MPC may compute another values which is smaller e.g. 35 computation time units.

In that case, if we specified a rate of change = 50, then the second $tC_{Control}$ value (i.e. 35) would not have been possible to be computed by the MPC. This is because the difference (rate of change) from 150 to 35 is larger than 50 exceeding the allowed rate change for the controller. In fact, the most allowed reduction from 150 would have been 100 (i.e. 150 - 50 = 100). This $tC_{Contorl}$ value can lead to admitting more jobs into the system or admitting jobs with large computation time value causing cluster overload (undesired) and potential *ntS* reduction (undesired).

- **Other Hard Constraints**: sometimes it can be impossible for the controller to satisfy all its hard constraints under all conditions. In such a case, the controller declares the optimal control QP problem to be infeasible i.e. an error condition.
- **Soft Constraints**: this test evaluates the constraint parameters to help achieve the proper balance of using hard and soft constraints. If a constraint is too soft, an unacceptable violation may occur. If it is too hard, the controller might pay it too much attention. Making a constraint harder cannot prevent a violation if the constraint is fundamentally infeasible.

Our MPC design passed all tests but with a warning for our MPC's closed-loop steady-state gains. The warning we received from the MPC toolbox was concerned with raising an awareness that one of the metrics weights are not maximised like its peers which could lead to controller under-performance for that metric. We opted for full weight values for the QoS levels of *PU* and ntS_{Design}. We chose a lower weight for ntS_{Data} QoS level because it suffers less *ntS* reduction than its Design type of peers hence we wanted to steer the MPC towards ntS_{Design}.

The Admission Control block deals with the two workflow tasks simultaneously. It compares tC_{Data} & tC_{Design} (tC values of newly arrived Data and Design tasks) with $tC_{ControlData}$ & $tC_{ControlDesign}$ (MPC's control signals for Data and Design tasks) respectively in real-time. The admission control logic here is to admit newly arrived tasks that do not exceed the control signal. If any task satisfies the admission logic (of its workflow type) then it is admitted for scheduling and processing. If not, then that particular task is believed to degrade the scheduling performance with respect to meeting the QoS levels and therefore is rejected.

The novel MPC-based FAC logic is:

```
if arrived task is Data
```

```
    if task computation time value <= control signal 4
        admit task;
    else
        kill task;
    end
elseif arrived task is Design
    if task computation time value <= control signal 5
        admit task;
    else
        kill task;
    end
end
```

In general control theory, MPC can deal with multi-objective optimisation problems [73]. We can notice in Figure 4.8-(b) that our MPC block sends two control signals addressing the two different task types (Data and Design). The interesting point here is that there is no dedicated control signal for achieving the *PU* QoS level as in the PID case, see Figure 4.7-(b). The reason is because our MPC maps multiple QoS levels in one control signal which gives it the advantage over the PID-based FAC algorithm. We will see how this can benefit our industrial WMS in meeting its RTS objectives in our evaluations in the next section.

4.4 Evaluations

We have tested the workloads depicted from Table 4.1 in a 10-processor HPC cluster using one of our industrial system's HPC test-beds size. Each case-study had 30 different trials, the plots shown in this section represent the average of those trials per case-study. One-way analysis of variance (ANOVA) tests were carried-out to statistically demonstrate the difference between the results obtained from experimental case-studies. Please note that our ANOVA tests relied on data normal distribution for the rest of evaluations in this thesis. It is worth mentioning that during all of our time-plots of our results in this thesis, we did not include the error marks (margins) because after taking the average of the experimental trials, we noticed the variance is trivial. Thus, we did not include them in our plots.

The reasons for the evaluations are to:

- Address the second research question in section 3.2 which is:
  "*Once we validate our system model, can we design an intelligent admission control algorithm for managing the trade-offs between the two RTS objectives without prior knowledge of jobs RTS parameters?*".

- Since PID control lacks performance in multi-objective optimisation problems, we want to evaluate its performance with simplistic task-sets of independent periodic tasks. Comparing our basic (PID) and advanced (MPC) novel FAC algorithms will help us decide at this stage which algorithm is better for our multi-objective RTS optimisation. So, we can choose the better algorithm as a basis for our further milestones in this thesis like handling aperiodic task-sets with dependencies, testing with a larger cluster size and different workload models; low, full and high.

For the experiments conducted in this section, the admission algorithms we tested are:

- Open-loop (OL), see Figure 4.3. It represents our industrial system without the FAC block.
- P, PI and PID-based FAC, see Figure 4.7-(b). It represents our industrial system supported with a basic control-theoretic admission controller.
- MPC-based FAC, see Figure 4.8-(b). It represents our industrial system supported with an advanced control-theoretic admission controller.

We will refer to each case-study by the admission algorithm (OL, P, PI, PID or MPC) concatenated with the scheduling policy experimented with. For instance, MPC-EDF refers to our MPC-based FAC algorithm with EDF.

For each RTS metric, we will review:

- RTS analysis: the performance of EDF against FCFS per algorithm.
- Admission Control analysis: the performance of all admission algorithms against each other. When we started evaluating the difference between the P, PI and PID-based FAC algorithms, we noticed that the first two were exactly the same in all case-studies whereas the full PID was less than 5% better in performance for each metric. Therefore, in order to avoid duplications in plots and analysis, we will include the analysis of the PID-based FAC algorithm and the rest of the P and PI results are tabulated in the Appendix in Table A.1.

4.4.1 Task Normalised Slack Values

For this metric, we expect:

- PID-based FAC to outperform the OL algorithm because PID is designed to meet the QoS levels or an acceptable balance between the two metrics altogether.

- MPC-based FAC to outperform the PID peer because MPC can address the multi-objective optimisation problem in one control signal per task type where PID cannot, refer to section 4.3 for more details.

For RTS analysis, we expect EDF to outperform FCFS, refer to Figure 4.9 for more insights and Table 4.3 for statistical findings:

- In the OL algorithm, EDF scored 3.71% higher ntS_{Design} values than FCFS. This is a small increase which can be explained by the high release rate of tasks into the system for scheduling and processing which made the scheduling policy less effective.

- In the PID-based FAC algorithm, EDF scored 12.7% higher ntS_{Design} values than FCFS.

- In the MPC-based FAC algorithm, EDF was 53.84% less than FCFS in ntS_{Design} values, which is counter-intuitive. However, the EDF-based algorithm did not violate the QoS levels i.e. it never scored less than -1. Additionally, this can be an example where we notice EDF's performance degradation in resource-constrained and cluster overload cases as argued by [73, 79]. This counter-intuitive result will be reviewed in more details in this section.

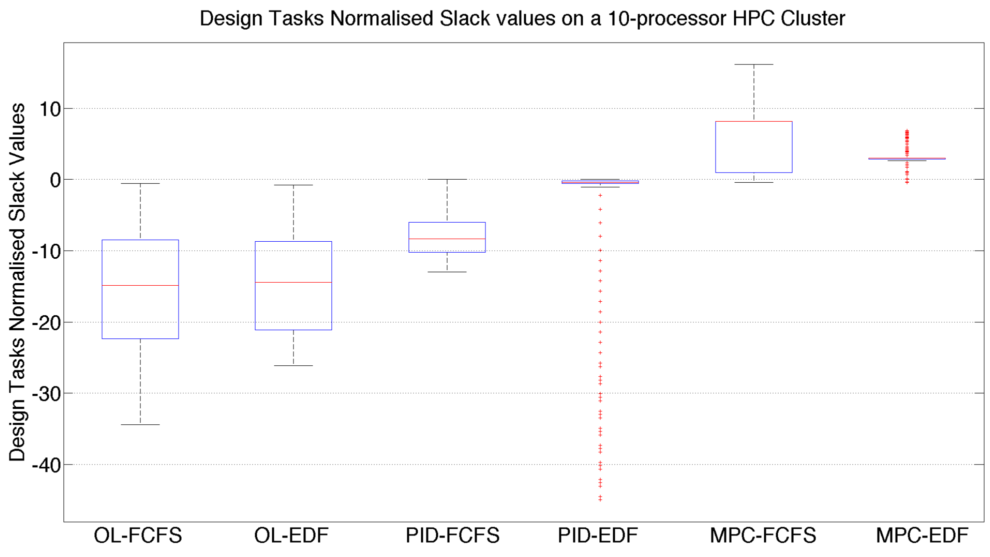

Figure 4.9: Design tasks normalised slack values.

If we want to compare the performance between FCFS and EDF in the MPC-based algorithm, we can review the time-plots for these two different algorithms with respect to (1) ntS_{Design}, (2) PU, (3) average queueing time for tasks and (4) the instances of killings conducted by the admission controller throughout the simulations (experiments), refer to Figure 4.10 and Figure 4.11 for more

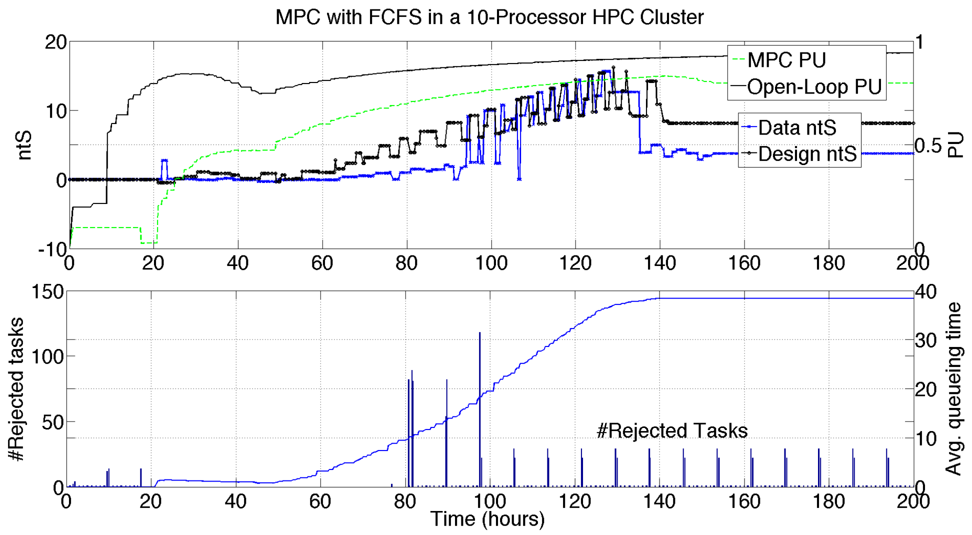

Figure 4.10: MPC-based FAC algorithms with FCFS in a 10-processor HPC cluster. The values for $Data_{ntS}$ and $Design_{ntS}$ in the top plots are for the MPC-based FAC algorithms.

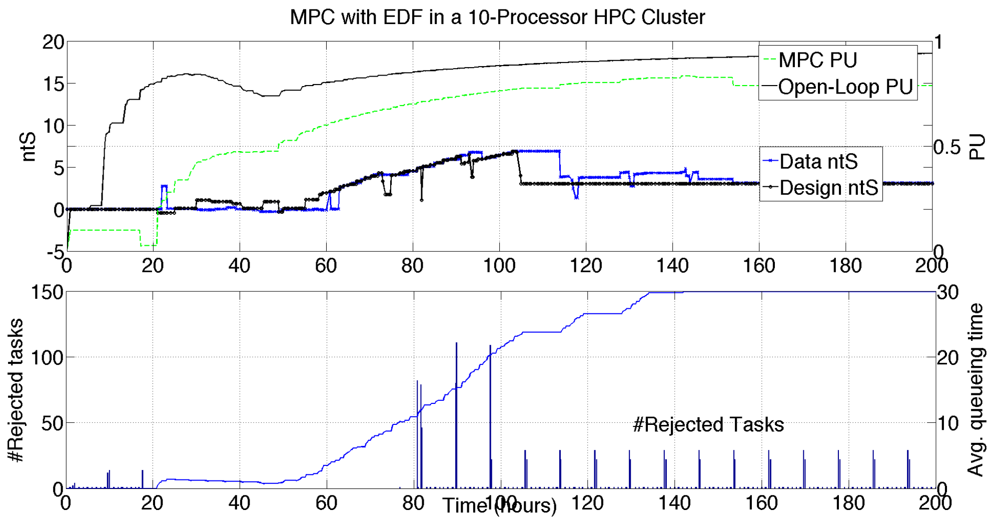

Figure 4.11: MPC-based FAC algorithms with EDF in a 10-processor HPC cluster. The values for $Data_{ntS}$ and $Design_{ntS}$ in the top plots are for the MPC-based FAC algorithms.

insights.

We notice from Figure 4.10 and Figure 4.11 that *PU* is similar between FCFS and EDF as statistical tests in Table 4.5 confirm so, yet the average queueing time is higher with EDF by 18.65%. Tasks killing instances (#Rejected Tasks) are overall similar in counts and periods except between the 80^{th} and 100^{th}-hour period as noticed in the graph, totalling to 1.32% higher killing rate with FCFS than EDF.

The higher the killing instances is due to FCFS's poor performance in meeting deadlines, hence the admission controller decides to remove more tasks in order not to violate the *ntS* QoS levels. Despite the small difference in killing instances, this can yield to a difference in *ntS* values as Table 4.3 confirms the significant difference between *ntS* values between FCFS and EDF. The higher the killing instances reduces the number of tasks to be processed in the cluster which leads to improved *ntS* values because there will be more processing capacity offered by the cluster.

As for the Data tasks, we expected Design tasks *ntS* values to be worse than their Data peers in the OL case as it was indicated for in our earlier system identification modelling, refer to subsection 4.1.4 for more details. As for the tabulated results for Data tasks, please refer to Table A.1 in the Appendix, where both types followed the same trend in which the OL algorithms scored the worst (least) *ntS* values. Then comes the PID-based FAC algorithm, and then the MPC peer scored the highest values for this metric.

From Table 4.3 we notice that two case-studies did not have significant difference between their means; *OL (FCFS vs. EDF)* and *PID (FCFS vs. EDF)* as their p-values were > 0.05. This reflects on the fact that when we experimented with the OL and the PID-based FAC algorithms, there was no significant difference between FCFS and EDF with respect to ntS_{Design}.

For admission control analysis, refer to Figure 4.9 for more insights and Table 4.4 for statistical findings:

- The OL algorithms scored the lowest ntS_{Design} values as expected. The PID-based FAC algorithm increased the metric by 42.85% and 49.5% more than the OL peers with FCFS and EDF, respectively. Yet, it suffered from violating the QoS level according to Figure 4.9.
- The MPC-based FAC algorithm scored 174.71% and 139.5% more than the PID peers with FCFS and EDF, respectively.

In conclusion to this metric, we notice that:

Case-Study (FCFS vs. EDF)	***p*-value**
OL	***0.24***
PID	***0.44***
MPC	$1.86e^{-38}$

Table 4.3: One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of ntS_{Design}.

Case-Study	***p*-value**
OL-FCFS vs. PID-FCFS	$1.54e^{-43}$
OL-EDF vs. PID-EDF	$2.49e^{-67}$
OL-FCFS vs. MPC-FCFS	$4.40e^{-201}$
OL-EDF vs. MPC-EDF	$1.27e^{-210}$
PID-FCFS vs. MPC-FCFS	$1.07e^{-242}$
PID-EDF vs. MPC-EDF	$4.95e^{-63}$

Table 4.4: One-way ANOVA test for the admission control experimental data of ntS_{Design}.

- The OL algorithms suffered the lowest ntS values.
- The PID-based FAC algorithm improved the system's performance in increasing ntS values, but it still suffered form negative values beyond the QoS level which is -1.
- The MPC-based FAC algorithm scored the highest ntS values and did not violate the QoS level for this metric.

4.4.2 Processor Utilisation

For this metric, we expect:

- The OL algorithms to outperform both of our FAC algorithms. This is because in OL there is no admission control which will naturally leave the cluster more utilised regardless of missing/meeting deadlines.
- We expect the MPC-based FAC algorithm to score higher PU, overall, than the PID peer. This is because MPC handled both metrics simultaneously whereas PID did not, refer to section 4.2 and section 4.3 for more details.

For RTS analysis, refer to Figure 4.12 for more insights and Table 4.5 for statistical findings. We notice that across all algorithms the difference in performance between FCFS and EDF is less than 5% in average. This can be explained by the high release rate of tasks which made the scheduling policy less effective. From Table 4.5, two case-studies did not have significant difference between their means; *OL (FCFS vs. EDF)* and *MPC (FCFS vs. EDF)* as their

p-values were > 0.05. This reflects on the fact that when we experimented with the OL and MPC-based FAC algorithms, there was no significant difference between FCFS and EDF with respect to PU.

For admission control analysis, refer to Figure 4.12 for more insights and Table 4.6 for statistical findings:

- The PID-based FAC algorithm scored less PU than the OL system by 15.83% and 18.49% with FCFS and EDF, respectively.
- The averaged PU value scored by the MPC-based FAC was less than the PID peer by 4.26% and 1.56% with FCFS and EDF, respectively.

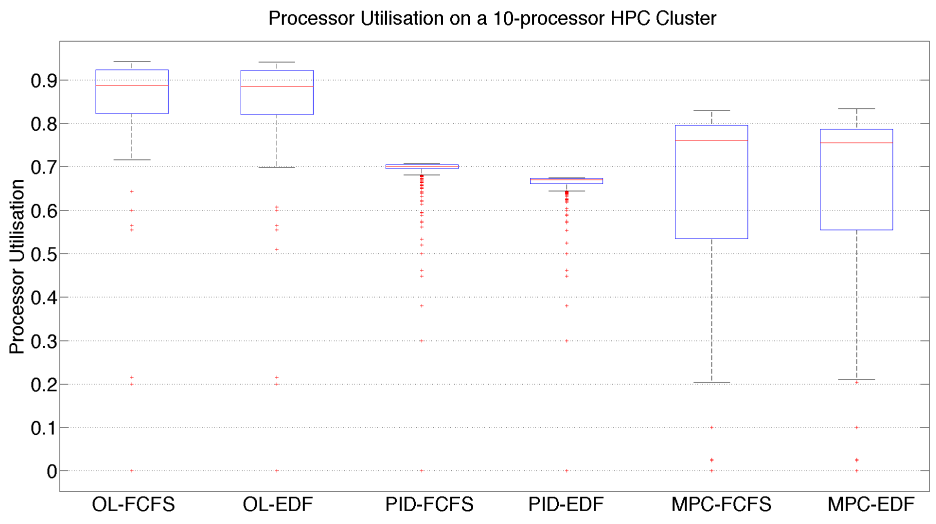

Figure 4.12: Processor utilisation.

In conclusion:

- Although the OL algorithms scored the highest PU, they experienced the lowest ntS values. This is not accepted because they lack managing the trade-offs between the two metrics with respect to their QoS levels.
- The PID-based FAC algorithm improved the system's performance but it still scored negative ntS values beyond the QoS level.
- The MPC-based FAC algorithm scored the highest ntS values and it did not violate the QoS level but with the expense of some PU loss which is expected due to the killing nature of the admission control.

Case-Study (FCFS vs. EDF)	***p*-value**
OL	***0.73***
PID	$2.78e^{-36}$
MPC	***0.80***

Table 4.5: One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of *PU*.

Case-Study	***p*-value**
OL-FCFS vs. PID-FCFS	0.00
OL-EDF vs. PID-EDF	0.00
OL-FCFS vs. MPC-FCFS	$1.19e^{-129}$
OL-EDF vs. MPC-EDF	$2.57e^{-137}$
PID-FCFS vs. MPC-FCFS	$5.54e^{-13}$
PID-EDF vs. MPC-EDF	$4.82e^{-31}$

Table 4.6: One-way ANOVA test for the admission control experimental data of *PU*.

4.5 Summary

In summary, due to MPC's balanced approach in managing the trade-offs between the RTS objectives, we propose the MPC-based FAC algorithm from this chapter as the answer to our second research question in chapter 3 which is: "*Once we validate our system model, can we design an intelligent admission control algorithm for managing the trade-offs between the two RTS objectives without prior knowledge of jobs RTS parameters?*". We propose MPC-based FAC for the next technical milestones to address aperiodic and dependent task-sets with a larger HPC cluster size of different workload models.

Chapter 5

Handling Dependencies in Admission Control Scheduling

The motivation for this chapter is to address our third research question which is concerned with managing the trade-offs between our RTS objectives when dealing with dependent task-sets. Dependencies can add a scheduling constraint to the admission decisions because dependencies enforce a particular execution order for dependent tasks. This can negatively affect any admission decision because we assume removing one task of a job will lead to discarding the whole job. This RTS challenge can be seen in industries where (1) their workloads include dependencies between tasks and (2) incomplete jobs are automatically discarded.

We will follow the design steps adopted in chapter 4. The reason for this is twofold. First, feedback control RTS algorithms follow these design steps. Second, our work in this chapter differs from our previous one in introducing dependencies between tasks. Because of that, the system dynamics will change at respecting task dependencies and further performance metrics have emerged as well. Therefore, it is meaningful to introduce the design steps followed in this chapter where, however, there will be similarities in some of the steps which will be highlighted accordingly.

Our contributions in this chapter are:

- Developing an enhanced version of our novel MPC-based FAC algorithm to address task dependencies as a scheduling constraint.
- Constructing a system model for capturing the RTS dynamics between jobs computation time values (input) and jobs normalised slack values & processor utilisation (outputs).
- Managing the trade-offs between our RTS objectives and other RTS

metrics that have emerged due to the notion of jobs, see section 5.3 for more details on the emerging RTS properties.

5.1 Task Model

Our previous chapter dealt with simple, independent task-sets. For this chapter, we used a validated synthetic workload generator resembling the workloads of our industrial system which is the work of Burkimsher in [16]. For each workload, the generator produces dependency patterns as exhibited in the industrial system which are a combination of the types in Figure 5.1. Typical industrial task dependency patterns resemble the ones obtained from Burkimsher's workload generator. Using this generator will eliminate the assumption most related works have implemented which is concerned with periodic jobs. This generator does not provide knowledge of jobs arrival times and rates apriori to the admission controller which is key to our work as mentioned in chapter 2.

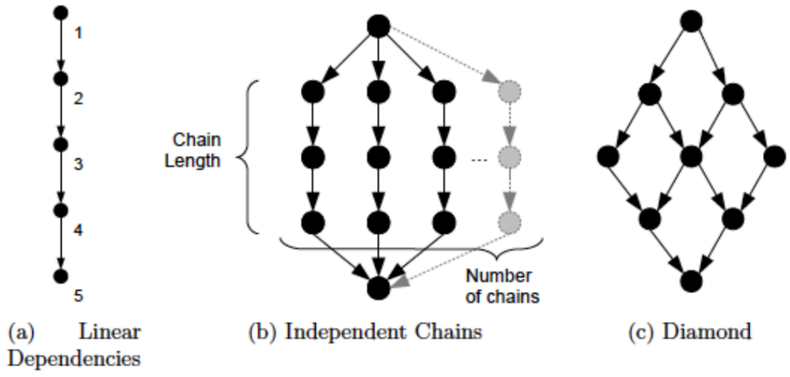

Figure 5.1: Tasks' dependency patterns [15].

Each workflow consists of around 200 aperiodic jobs and 1000 dependent tasks. It represents 300-500 hours (2-3 weeks) worth of our industrial system's RTS dynamics. We have three types of workloads with respect to target cluster PU: low, 80%; full, 100%; and high, 120%. We can summarise our RTS notations in Table 5.1. For each job, we take the longest-path, computation time-wise,

RTS Specification	RTS Notion
i^{th} Job ID	jID_i where $1 \leq i \leq 200$
i^{th} Job Critical Path	jCP_i
i^{th} Job Normalised Slack	njS_i
i^{th} Task ID	tID_i where $1 \leq i \leq 1000$
i^{th} Task Computation Time	tC_i where 1 hours $\leq tC_i \leq$ 150 hours
i^{th} Task Deadline	tD_i
i^{th} Task Normalised Slack	ntS_i

Table 5.1: Task-set's RTS specifications [41].

as its critical path jCP_i and deadline value; $jC_i = jD_i$. In the general context, the notion of a critical path is unitless as it refers to the sequence of tasks within a job that forms the longest path for the individual job. However, in this thesis, we use the term jCP_i as the job's longest path computation time value which will be used in calculating slack values in the coming equations.

Tasks can be categorised into critical and non-critical i.e. whether belonging to the job's critical path or not. For critical tasks, a task's tD is computed as the summation of all other critical tasks' tC values from the start of the job until the task we are evaluating, see Equation 5.1.

$$tD_i = \sum tC_{1:i} \tag{5.1}$$

For non-critical tasks, we select the task's shortest path (tSP), computation time-wise, that it falls on. The tD is then computed with respect to the (1) summation of tasks' tC values from the start of tSP including the one we are evaluating, $\sum tC_{1:i}$, (2) tSP value and (3) the job's jCP value, see Equation 5.2 and Figure 5.2.

$$tD_i = \frac{\sum tC_{1:i}}{tSP_i} * jCP_i \tag{5.2}$$

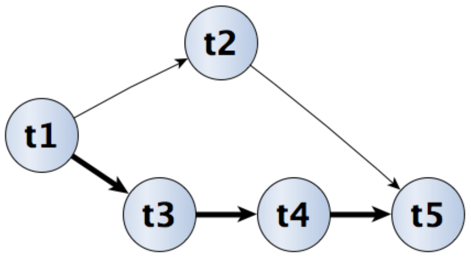

Figure 5.2: An example for calculating deadline values for critical and non-critical tasks. The task path with thick arrows represent the critical tasks forming the job's critical path.

We can consider the job in Figure 5.2 represented by its tasks. For ease of reference and calculations in this example only, we take Task IDs to be equal to their computation time values i.e. $tID_3 = tC_3 = 3$ and so on. We can notice that tasks $[t_1, t_3, t_4$ and $t_5]$ are the critical tasks (because they compose the longest path computation time-wise) while t_2 is non-critical. For the critical tasks, we apply Equation 5.1:

- $tD_1 = 1$;
- $tD_3 = 1 + 3 = 4$;
- $tD_4 = 1 + 3 + 4 = 8$;

- $tD_5 = 1 + 3 + 4 + 8 = 13$; This is the critical path value for the job (jCP).

For the the non-critical tasks, we apply Equation 5.2:

- $tD_2 = \frac{(1+2)}{(1+2+4+5)} * 13 = \frac{13}{4} = 3.25$;

Similarly to chapter 4, we use the notion of tS as defined in chapter 3, see Equation 3.2. We normalise tS with respect to the task's tD, see Equation 4.1. Similarly to tasks, jobs jS values include monitoring how far a job is from its completion, i.e. its last task's completion, once released as a whole by the end-user until its last task leaves the cluster, see Equation 3.3. We normalise jS values with respect to the job's jCP value, see Equation 5.3.

$$njS_i = \frac{jCP_i - Time - jCP_i'}{jCP_i} \tag{5.3}$$

jCP_i' is the i^{th} job's remaining computation time of the critical path.

5.2 Scheduling Policy

This remains the same as in chapter 4.

5.3 Performance Metrics

- PU, see Equation 3.1.
- njS, see Equation 5.3.

Regulating the workloads into the cluster may lead to admission decisions such as killing one or more tasks (and jobs). For instance, if a decision has been made to kill a task which belongs to a job that has had half of its tasks already processed in the cluster, the job will, in theory, be killed, and all future tasks belonging to this very job will not be admitted. They will not be admitted because they belong to an incomplete job, and this may lead to wasted processor utilisation (WPU). We call it "wasted" because the time the cluster spent processing half of that job, has not been positively utilised for completing the whole job. This example can also lead us to another RTS metric regarding the admission decisions, jobs killing ratio (jKR). This can be analysed as a ground for justifying the admission decisions in order to meet our RTS objectives. Our additional RTS performance metrics are:

- WPU, we compute the time the cluster spent processing jobs that are killed against the time the cluster spent processing all jobs, see Equation 5.4.

$$WPU = \frac{\sum Time_{Killed}}{\sum Time_{All}} \tag{5.4}$$

- jKR, we take the ratio between killed and total released number of jobs, see Equation 5.5.

$$jKR = \frac{\#Jobs_{Killed}}{\#Jobs_{All}} \tag{5.5}$$

We desire managing the trade-offs, in real-time, between maximising both njS and PU. Additional RTS objectives for this chapter include scoring minimal WPU and jKR.

5.4 System Model and Identification

Due to task dependencies, we are now dealing with the notion of jobs. Our MPC-based FAC algorithm, proposed in chapter 4, is blind to the notion of jobs. So, we had to create a system model to capture correlations (system dynamics) between jCP (input) and PU & njS (outputs). We followed the same identification steps in chapter 4 for obtaining system models for tasks and jobs. However, we took the system inputs (tC and jCP) from Burkimsher's generator, refer to Table 5.1, then were simulated in the OL simulation model[1]. We simulated the inputs for approximately 200 hours in order to capture correlations (system dynamics) between inputs and outputs for tasks ((tC and PU & ntS) and for jobs (jCP and PU & njS).

As for the tasks and jobs system models' accuracy, they scored 85.29% and 80.08%, respectively, where their error plots are in Figure 5.3. The models we obtained are placed in the System Model block in Figure 5.4 and each MPC block refers to its relevant system model accordingly.

5.5 Closed-Loop Scheduling Control

In this section, we will review our enhanced MPC-based FAC algorithm and the other admission control baselines used for evaluation purposes.

5.5.1 A Novel MPC-based FAC Algorithm for Dependent Tasks

Our proposed algorithm here resembles the previous one in chapter 4 but with extra Admission Control and MPC blocks for handling jobs. The Jobs MPC block resembles Tasks MPC in design, but it takes njS values, instead of ntS, as part of its computations. The Jobs AC block compares newly arrived jobs jC values against the control signal, $jC_{Control}$, which is the admission threshold, received from the Jobs MPC block. The reason for adding the two components for jobs is to enable our proposed algorithm to perform admission decisions on

[1]The OL simulation model is in principle the closed-loop system but without the AC and MPC blocks present, see Figure 5.4.

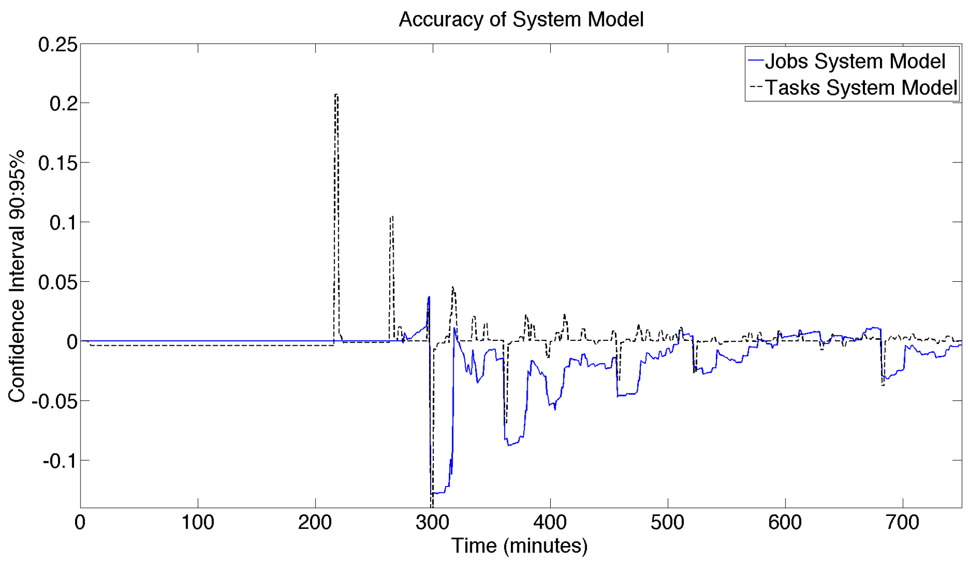

Figure 5.3: Tasks and jobs system identification model error: measured minus simulated data. The reason the x-axis is not the full 200 hours is because in system identification we allocate a small part of the measured data for the model estimation and the rest for model validation [46].

the jobs-level to reduce WPU and jKR. The motivation here is for the Jobs MPC block to predict early enough if a job will miss its deadline given the HPC capacity. If so, then the Admission Control block will remove this job as soon as one of this job's tasks arrives after passing through the Dependency Check block.

The challenge here is removing the job early enough without causing WPU and of course making sure we are removing the right job judging on its jC value with respect to $jC_{Control}$. We will refer to this algorithm as AC_jAC_t for the rest of this thesis, see Figure 5.4. As for the previous algorithm, it will be referred to as AC_t signifying its tasks-level admission control. We removed the term MPC in the algorithm's name in order to save repeating it in our FAC algorithms.

As for the controllers' stability, we tested our Tasks and Jobs MPC controllers inside MPC toolbox in Matlab environment, similarly to chapter 4. They passed all tests with a warning in regards to each controller's steady-state gains suggesting sacrificing one output's weight values to zero in order to reduce possible errors in other outputs. However, we opted for full weight values for all system outputs because they are all equally important to our RTS optimisation problem.

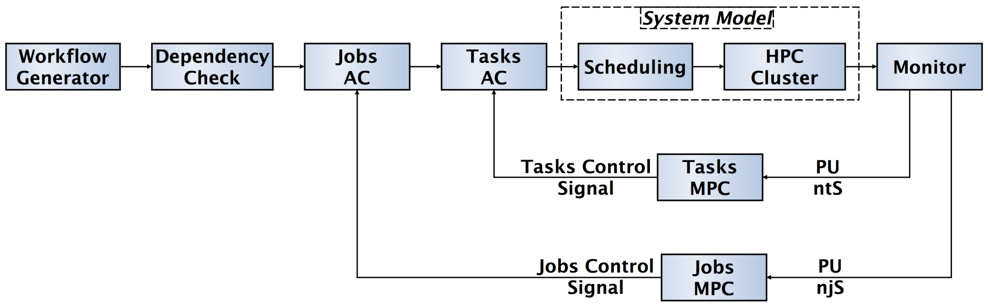

Figure 5.4: A block diagram of our new, novel MPC-based FAC algorithm proposed for handling dependent real-time tasks referred to as AC_jAC_t due to the double Admission Control blocks.

5.5.2 Baseline Admission Control Algorithms

The purpose of presenting baseline admission control algorithms is to have baselines for comparative analysis for our proposed, novel AC_jAC_t algorithm. As our previous algorithm, AC_t, was concerned with tasks admission only, our new algorithm does admission on the jobs-level as well. Thus, we find it necessary to compare baseline algorithms that, also, practise admissions on the job's level, see Figure 5.5.

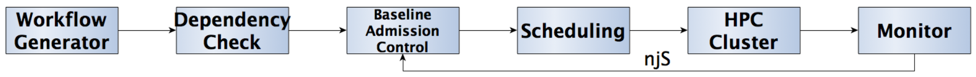

Figure 5.5: A block representation of our baseline algorithms.

The algorithms are triggered when njS is < 0, i.e. system is running late in jobs processing. They only deal with live jobs. "Liveness" here refers to jobs that are released to the system for scheduling and processing, but have not completely left the cluster yet i.e. some of its tasks are still in the system for scheduling and/or processing.

Random Admission Control, AC_{Random}

The idea here is to implement admission decisions based on killing a random number, N, out of live jobs only when the system experiences negative njS values. For instance, if we have five live jobs and the system experiences negative njS values, the AC_{Random} looks at the five jobs by large and computes N out of the total number of live jobs. N can be an integer between one and five. N is defined so that it increases njS values, desirably ≥ 0, while not leading to reduction in PU i.e. killing N jobs that preferably have not yet started processing so not to directly affect PU, refer to Figure 5.6. However, we will notice with higher workloads that AC_{Random} decided to kill further jobs for the sake of increasing njS.

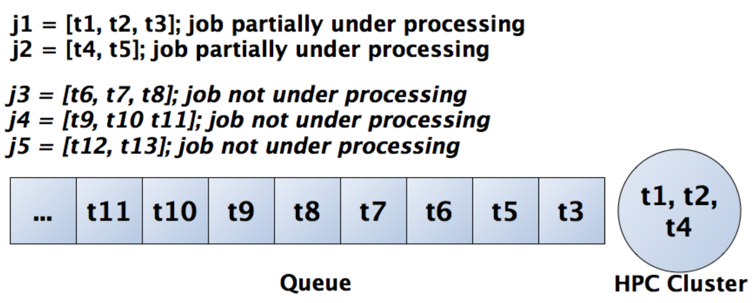
j1 = [t1, t2, t3]; job partially under processing
j2 = [t4, t5]; job partially under processing
j3 = [t6, t7, t8]; job not under processing
j4 = [t9, t10 t11]; job not under processing
j5 = [t12, t13]; job not under processing
... t11 t10 t9 t8 t7 t6 t5 t3
t1, t2, t4
Queue
HPC Cluster

Figure 5.6: An illustration for admission decision-making of AC_{Random} . The jobs highlighted in italic are the ones which AC_{Random} can consider for killing because they have not started processing in the cluster. The other jobs i.e. $j1$ and $j2$ are outside the decision-making range for AC_{Rand} because they are partially i.e. some of their tasks are being processed in the cluster and hence removing those jobs can decrease PU and also cause WPU as both consequences are undesired.

Job Slack Admission Control, AC_{jSlack}

This algorithm practises jobs admission by monitoring live jobs' njS values. It decides to kill the job(s) that are running late the most i.e. those with the least njS values. Using the example mentioned in the previous baseline, this algorithm removes the late jobs out of the five live ones.

Job Completion Ratio Admission Control, AC_{jCR}

This one is concerned with removing live jobs that have the least completion ratio. The idea here is when the system experiences negative njS values, we can remove jobs with the least completion ratio because the system is already late and a lot of processing (computation time) remains ahead of those jobs. Using the example used in the previous baselines, out of the five live jobs, this algorithm removes late jobs that have been processed by the cluster the least.

5.6 Evaluations

The workflows depicted from Table 5.1 using Burkimsher's generator were tested in a 50-processor HPC cluster. Each workflow contains jobs being generated at least three times, we ran each case-study 10 times and we averaged the simulation results in the metrics plots. One-way analysis of variance (ANOVA) tests were carried-out to statistically demonstrate the difference between the results obtained from experimental case-studies. We will refer to each case-study by the admission algorithm (OL, AC_t, AC_jAC_t, AC_{Random}, AC_{jSlack} and AC_{jCR}) concatenated with the scheduling policy experimented with. For instance, AC_jAC_t-EDF refers to our proposed algorithm with EDF.

For each RTS metric, we will review:

- RTS analysis: the performance of EDF against FCFS per algorithm.
- Admission Control analysis: the performance of all systems against each other. When we started evaluating the difference between the baseline algorithms performances, we noticed that AC_{jSlack} scored the same values as AC_{jCR}. We observed that when the system experiences negative njS values, all late jobs (with least slack values) were those ones that have been processed the least. Therefore, in order to avoid duplications in plots and analysis, we will refer to both admission control baselines as $AC_{jSlack/CR}$.

Please note that full numeric results for all workload types (low, full and high) are tabulated in the Appendix in Table B.1. The high workload cases are reviewed in this section due to their high RTS dynamics that will reflect more on the different performances of the algorithms. We will also include the notion of simulation time ($SimTime$) in our evaluations. $SimTime$ represents the length of the simulation in time units (hours) each algorithm has scored i.e. the time period each algorithm spent in scheduling and processing all arriving jobs.

The reason for including $SimTime$ is because it provides us with different insights than PU's. For instance, an algorithm with 500 hours $SimTime$ and 60% average PU is different from another algorithm that scored 150 hours $SimTime$ with the same PU. This can be an indication that the former algorithm with a larger $SimTime$ has processed a higher number of jobs and/or larger jobs computation time-wise. This is reflected on the longer simulation time ($SimTime$) which is desired. We will see how this varies for our RTS metrics.

5.6.1 Job Normalised Slack Values

For this metric, we expect:

- The OL algorithms to score the lowest njS values because they do not manage the trade-offs between the RTS objectives in addition to their poor scheduling performance during cluster overload situations.
- AC_t to improve the performance in comparison to the OL but will still violate the QoS levels because it is blind to the notion of jobs, as seen in chapter 4.

- Our novel AC_jAC_t to score the highest njS values due to its approach in balancing both RTS objectives in real-time as opposed to the other baseline algorithms.

For RTS analysis, we expect EDF to outperform FCFS with respect to maximising njS values, refer to Figure 5.7 for more insights and Table 5.2 for statistical findings:

- In the OL algorithm, EDF increased njS by 163.06% more than FCFS. Both policies scored negative njS values beyond the QoS level.

- In AC_t, EDF increased njS by 14.93% more than FCFS. Although AC_t is blind to the notion of jobs, when experimented with EDF, it scored ≥ 0 njS values, which is a good result. However, we will see this good result was achieved at the expense of the other RTS metrics which will be reviewed in the next sections.

- In AC_jAC_t, EDF increased njS by 4.37% more than FCFS. This small difference can be explained by the high workload the admission controller was receiving which made the scheduling policy less effective. Both policies scored ≥ 0 njS values, which is a good result and expected due to AC_jAC_t jobs admission level.

- In AC_{Random}, EDF decreased njS by 8.73% than FCFS. This counter-intuitive result can be explained by the randomised admission logic of AC_{Random} which can lead to scenarios where a random killing decision can remove a not-so-late job resulting in further decreasing njS (i.e. increasing lateness) of already late jobs regardless of the adopted scheduling policy. We will see this effect in other RTS metrics in the next sections as well. Both policies scored negative njS values beyond the QoS level.

- In $AC_{jSlack/CR}$, EDF increased njS by 4.17% more than FCFS. Both policies scored negative njS values beyond the QoS level.

We notice that EDF is better than FCFS across all of the admission control algorithms, except for AC_{Random}, which confirms our expectations in the benefit of introducing EDF with respect to maximising njS. This good performance is expected due to EDF advancing jobs with shorter deadlines ahead in the queue resulting in overall higher njS values.

For admission control analysis, refer to Figure 5.7 for more insights and Table 5.3 for statistical findings:

- AC_t-FCFS increased njS by 288.22% more than OL-FCFS.

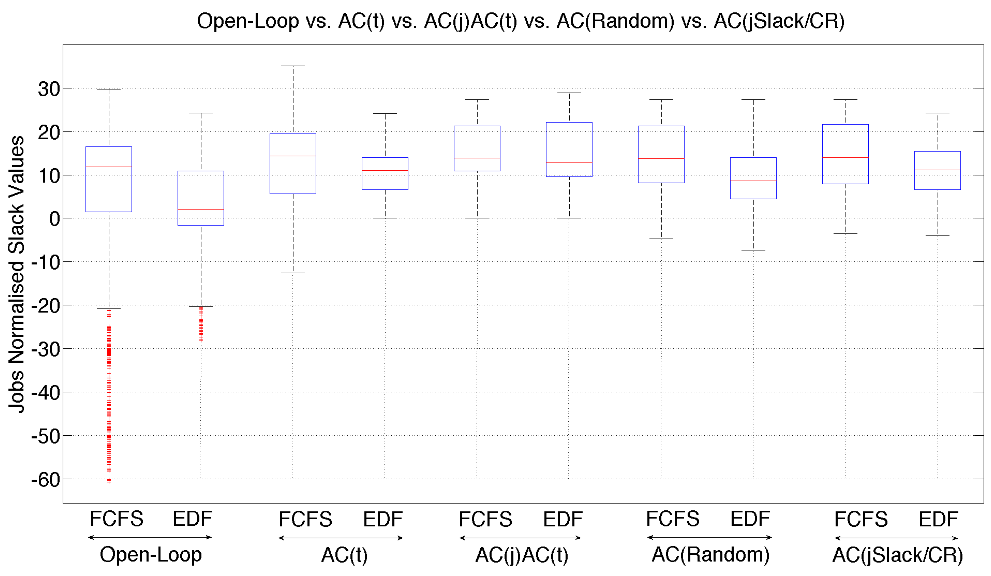

Figure 5.7: Jobs normalised slack values box-plots of high workloads, 120%.

- AC_t-EDF increased njS by 69.61% more than OL-EDF. The percentages of improvement in the FCFS case is larger than the ones associated with EDF. This is because EDF is already better than FCFS in the OL system.
- AC_jAC_t-FCFS increased njS by 12.55% more than AC_t-FCFS.
- AC_jAC_t-FCFS increased njS by 4.97% more than AC_{Random}-FCFS.
- AC_jAC_t-FCFS decreased njS by 1.44% less than $AC_{jSlack/CR}$-FCFS. This counter-intuitive result can be explained by $AC_{jSlack/CR}$'s higher jKR than our AC_jAC_t, see subsection 5.6.3 for more details. The higher the jKR the higher the njS, but we desire ≥ 0 njS with minimum jKR. We also notice that $AC_{jSlack/CR}$ violated the QoS levels for this metric by scoring beyond -1 njS. They were close in performance to our proposed algorithm due to their admission logic in removing the latest and least completed jobs, respectively, when the system experiences negative njS values. However, AC_{jSlack} and AC_{jCR} suffered the lowest $SimTime$, which is undesired, refer to Table 5.6 for more details.
- AC_jAC_t-EDF increased njS by 2.21% more than AC_t-EDF.
- AC_jAC_t-EDF increased njS by 19.23% more than AC_{Random}-EDF.
- AC_jAC_t-EDF decreased njS by 7.35% more than $AC_{jSlack/CR}$-EDF. However, $AC_{jSlack/CR}$ violated the QoS level in addition to their $SimTime$ being lower than AC_jAC_t-EDF by 226.2 hours reflecting on $AC_{jSlack/CR}$'s high jKR values, refer to Table 5.6 and subsection 5.6.3 for more details.

Due to the logic which each admission control baseline follows, the more late the system becomes, the higher the probability each baseline algorithm will

Case-Study (FCFS vs. EDF)	***p*-value**
OL	$9.26e^{-143}$
AC_t	$3.16e^{-34}$
AC_jAC_t	$0.05e^{-2}$
AC_{Random}	$4.83e^{-212}$
$AC_{jSlack/CR}$	$5.16e^{-89}$

Table 5.2: One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of njS.

Case-Study	***p*-value**
OL-FCFS vs. AC_t*-FCFS*	$4.66e^{-81}$
OL-EDF vs. AC_t*-EDF*	0.00
AC_t*-FCFS vs.* AC_jAC_t*-FCFS*	$3.72e^{-18}$
AC_t*-EDF vs.* AC_jAC_t*-EDF*	$6.26e^{-112}$
AC_jAC_t*-FCFS vs.* AC_{Random}*-FCFS*	$2.73e^{-5}$
AC_jAC_t*-EDF vs.* AC_{Random}*-EDF*	$8.66e^{-283}$
AC_jAC_t*-FCFS vs.* $AC_{jSlack/CR}$*-FCFS*	***0.17***
AC_jAC_t*-EDF vs.* $AC_{jSlack/CR}$*-EDF*	$2.15e^{-78}$

Table 5.3: One-way ANOVA test for the admission control experimental data of njS.

decide in removing further jobs from the system according to its logic for the sake of increasing njS values. Thus, the more the killing, the higher the opportunity for more jobs to meet their deadlines and hence having increased njS values.

In conclusion to this metric we notice:

- EDF is better than FCFS in maximising njS.
- Adding an admission controller decreases the difference between the two scheduling policies with respect to maximising njS. This is due to the benefits of killing jobs with respect to maximising njS values regardless of the adopted scheduling policy.
- AC_jAC_t is the only algorithm that scored ≥ 0 njS with both FCFS and EDF.

5.6.2 Processor Utilisation

For this metric, we expect:

- The OL algorithm to score the highest PU because it does not practice any admission control leaving the cluster the busiest regardless missing/meeting deadlines.

- AC_t and AC_{Random} to score the least PU since both can unnecessarily kill jobs as the former is blind to the notion of jobs and the latter due to its random admission process.
- AC_jAC_t to score the closest to the OL algorithm because it manages both RTS objectives dynamically in real-time.

For RTS analysis, we expect EDF to outperform FCFS with respect to maximising PU, refer to Figure 5.8 for more insights and Table 5.4 for statistical findings:

- In the OL algorithms, EDF increased PU by 16.93% more than FCFS.
- In AC_t, EDF increased PU by 16.1% more than FCFS.
- In AC_jAC_t, EDF increased PU by 20.43% more than FCFS.
- In AC_{Random}, EDF decreased PU by 19.87% than FCFS. The counter-intuitive result stems from the random admission mechanism this baseline follows in potentially killing larger jobs (jC-wise) unnecessarily resulting in PU loss.
- In $AC_{jSlack/CR}$, EDF increased PU by 10.13% more than FCFS.

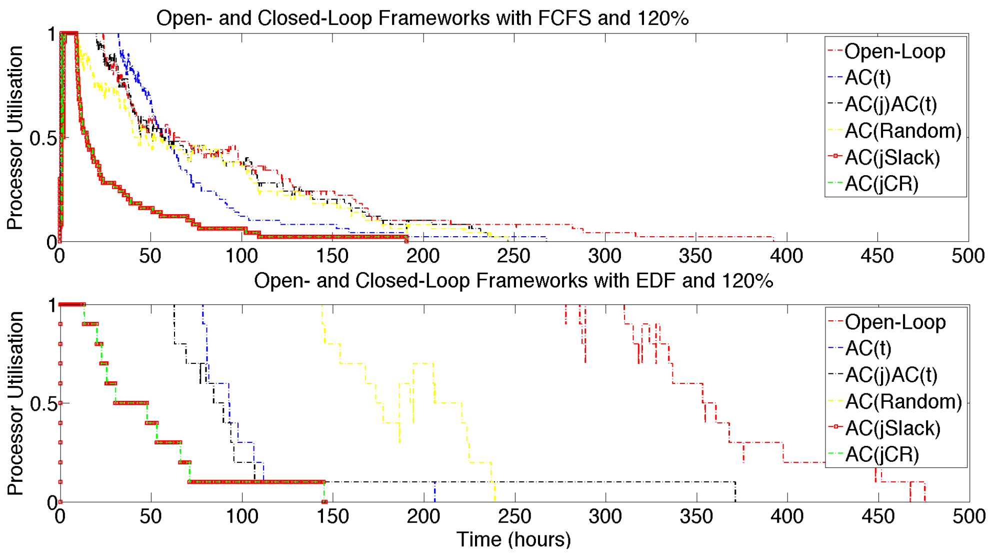

Figure 5.8: Processor utilisation time-plots of high workloads, 120%.

$AC_{jSlack/CR}$ had the smallest $SimTime$ which means they have processed a lesser number of jobs and/or smaller jobs (jC-wise) that have utilised the cluster for around 190 hours only. Thus, taking the average PU value during that period may result to a high value, but in comparison to other algorithms' $SimTime$, we can notice $AC_{jSlack/CR}$ were the poorest. In contrast, AC_jAC_t-EDF kept the cluster the busies the most amongst all admission

Case-Study (FCFS vs. EDF)	***p*-value**
OL	0.00
AC_t	$5.56e^{-233}$
AC_jAC_t	0.00
AC_{Random}	0.00
$AC_{jSlack/CR}$	0.00

Table 5.4: One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of PU.

Case-Study	***p*-value**
OL-FCFS vs. AC_t*-FCFS*	$9.12e^{-5}$
OL-EDF vs. AC_t*-EDF*	$3.91e^{-17}$
AC_t*-FCFS vs.* AC_jAC_t*-FCFS*	$8.74e^{-15}$
AC_t*-EDF vs.* AC_jAC_t*-EDF*	***0.06***
AC_jAC_t*-FCFS vs.* AC_{Random}*-FCFS*	$0.02e^{-2}$
AC_jAC_t*-EDF vs.* AC_{Random}*-EDF*	$2.3e^{-14}$
AC_jAC_t*-FCFS vs.* $AC_{jSlack/CR}$*-FCFS*	***0.76***
AC_jAC_t*-EDF vs.* $AC_{jSlack/CR}$*-EDF*	0.01*

Table 5.5: One-way ANOVA test for the admission control experimental data of PU. Mark (*) indicates that the difference between the data-sets is almost insignificant.

control algorithms, see Table 5.6 for more details.

For admission control analysis, refer to Figure 5.8 for more insights and Table 5.5 for statistical findings:

- AC_t-FCFS scored lower PU by 0.22% than OL-FCFS.
- AC_t-EDF scored lower PU by 0.61% than OL-EDF. The small difference between the two algorithms can reflect on the good performance AC_t has achieved in increasing njS with minimal PU loss. However, we can notice in Table 5.6 that AC_t scored a lesser $SimTime$ than the OL algorithm due to the former's tasks killing decisions resulting in removing jobs that contain those killed tasks leading to a much lesser $SimTime$ overall.
- AC_jAC_t-FCFS scored lower PU by 3.23% than AC_t-FCFS. Although the average PU of our enhanced algorithm is lesser than the previous one, a closer look into the $SimTime$ performance in Table 5.6 will reveal to us that our enhanced algorithm has managed to keep the cluster busy with <8% difference in averages.
- AC_jAC_t-FCFS scored higher PU by 2.13% than AC_{Random}-FCFS.
- AC_jAC_t-FCFS scored higher PU by 1.34% than $AC_{jSlack/CR}$-FCFS.

- AC_jAC_t-EDF scored higher PU by 1.1% than AC_t-EDF.
- AC_jAC_t-EDF scored higher PU by 2.69% than AC_{Random}-EDF.
- AC_jAC_t-EDF scored higher PU by 11.64% than $AC_{jSlack/CR}$-EDF.

All admission control algorithms share similar performances with respect to their PU average values due to the shared principle of killing jobs from the system in order to improve the performance with respect to our RTS objective. However, AC_jAC_t-EDF was the second longest $SimTime$ after the OL system and it outperformed the other baseline admission algorithms in both PU and $SimTime$.

Case-Study	$SimTime$ **(hours)**
OL-FCFS	393.00
OL-EDF	476.00
AC_t-*FCFS*	267.60
AC_t-*EDF*	205.90
AC_jAC_t-*FCFS*	247.00
AC_jAC_t-*EDF*	371.20
AC_{Random}-*FCFS*	246.50
AC_{Random}-*EDF*	238.80
$AC_{jSlack/CR}$-*FCFS*	191.00
$AC_{jSlack/CR}$-*EDF*	145.00

Table 5.6: Simulation time ($SimTime$) of all algorithms. $SimTime$ indicates how long each algorithm spent in scheduling and processing all arriving jobs.

In conclusion to this metric we notice:

- EDF is better than FCFS in maximising PU.
- OL-EDF is a better algorithm for maximising PU. AC_jAC_t-EDF is better for PU and njS altogether i.e. positive njS values with a small difference in PU in comparison to the other admission control algorithms.
- AC_jAC_t-EDF scored the second highest $SimTime$ after the OL peer.

5.6.3 Job Killing Ratio and Wasted Processor Utilisation

For these metrics, we expect:

- A correlation between the metrics where the higher the jKR the higher the WPU. This is because the more we kill jobs the higher the opportunity of killing one that have already started processing in the cluster resulting in some WPU.

- AC_t and AC_{Random} to score the highest jKR because they both may kill jobs unnecessarily.
- $AC_j AC_t$ to score the least jKR because it will only kill a job if it exceeds the admission threshold, $jC_{Control}$, which is mapped to both RTS objectives in real-time.

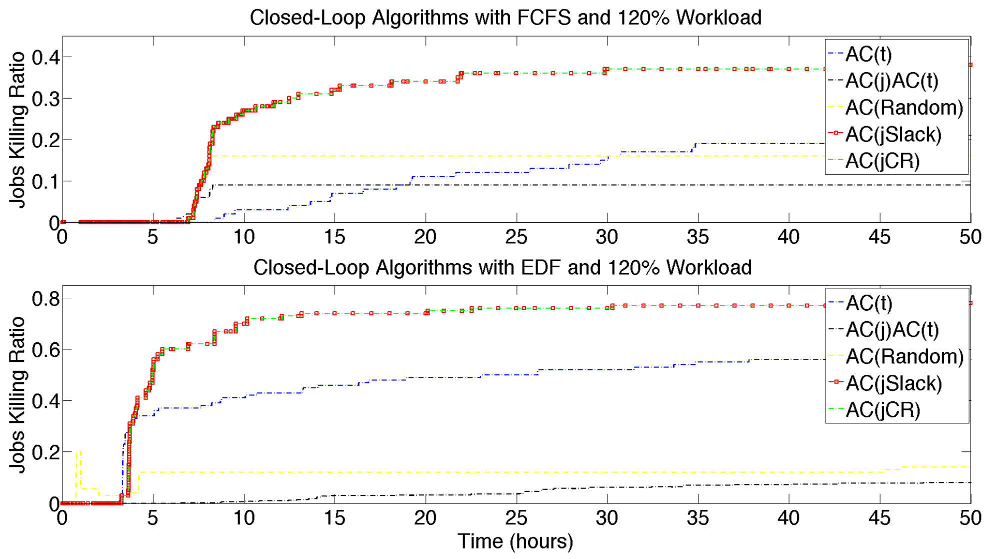

Figure 5.9: Jobs killing ratio time-plots of high workloads, 120%. Please note that we zoomed the time-plots to include the first 50 hours of the simulations because it is during this period the cluster was found to be fully utilised.

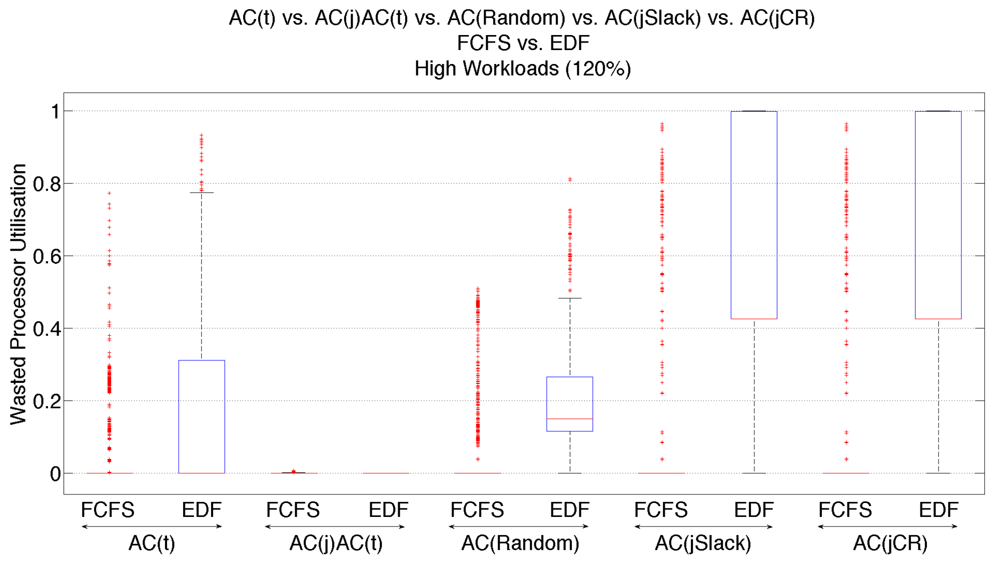

Figure 5.10: Wasted processor utilisation box-plots of high workloads, 120%.

For RTS analysis, refer to Figure 5.9 and Figure 5.10 for more insights and refer to Table 5.7 for statistical findings:

- In AC_t, EDF increased jKR by 112.12% while decreased WPU by 20.77%. The EDF-based algorithm increased jKR due to the fact AC_t

Case-Study (FCFS vs. EDF)	jKR **p-value**	WPU **p-value**
AC_t *(FCFS vs. EDF)*	$4.98e^{-32}$	$1.81e^{-35}$
AC_jAC_t *(FCFS vs. EDF)*	$3.03e^{-31}$	0.00
AC_{Random} *(FCFS vs. EDF)*	0.00	0.00
$AC_{jSlack/CR}$ *(FCFS vs. EDF)*	$2.62e^{-37}$	0.00

Table 5.7: One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of jKR and WPU.

Case-Study	jKR **p-value**	WPU **p-value**
AC_t*-FCFS vs.* AC_jAC_t*-FCFS*	***0.11***	$5.92e^{-8}$
AC_t*-EDF vs.* AC_jAC_t*-EDF*	$1.42e^{-74}$	0.00
AC_jAC_t*-FCFS vs.* AC_{Random}*-FCFS*	$0.03e^{-1}$	$2.07e^{-5}$
AC_jAC_t*-EDF vs.* AC_{Random}*-EDF*	0.00	0.00
AC_jAC_t*-FCFS vs.* $AC_{jSlack/CR}$*-FCFS*	$4.3e^{-49}$	$5.47e^{-41}$
AC_jAC_t*-EDF vs.* $AC_{jSlack/CR}$*-EDF*	$1.5e^{-14}$	0.00

Table 5.8: One-way ANOVA test for the admission control experimental data of jKR and WPU.

does admission decisions on the tasks-level. Therefore every time a task is removed by this algorithm, the whole job is consequently removed. Although EDF doubled jKR but it reduced WPU from the FCFS-based algorithm.

- In AC_jAC_t, EDF increased jKR by 13.13% while decreased WPU by 100%. The EDF-based algorithm incurred an increase in its jKR but scored 0 WPU in comparison to the FCFS-based peer.

- In AC_{Random}, EDF increased jKR by 82.24% and increased WPU by 81.7%. Due to the random logic of this admission controller, we do not expect a causal correlation between its FCFS- and EDF-based versions.

- In $AC_{jSlack/CR}$, EDF increased jKR by 88.37% while decreased WPU by 44.95%.

In conclusion to the RTS analysis, EDF scored higher jKR than FCFS in all admission control algorithms. The reasons vary depending on the admission controller's mechanism/logic. Despite the increase in jKR, EDF-based admission control reduced WPU in all algorithms. This can be explained by the lack of performance of EDF in cluster-overload situations as pointed out by Maggio et al. in [79] resulting in higher jKR. Yet, it has managed removing jobs from the system early enough so it did not cause as much WPU as FCFS.

For admission control analysis, refer to Figure 5.9 and Figure 5.10 for more insights and refer to Table 5.8 for statistical findings:

- AC_jAC_t-FCFS outperformed AC_t-FCFS in decreasing jKR by 72.72% and WPU by nearly 100%. According to our statistical tests in Table 5.8, the difference between the jKR data-sets was insignificant. This can be explained by the fact we compared the end values of jKR and WPU for all algorithms here, and the difference between such end-values was large, refer to subsection 5.6.4 for more details on the killing instances. However, the time-plots show close performance between the two algorithms here in Figure 5.9 where due to AC_t's tasks-level admission, it killed tasks at later periods of the simulation leading to larger differences at the end.

- AC_jAC_t-EDF outperformed AC_t-EDF in decreasing jKR by 85.2% and WPU by 100%.

- AC_jAC_t-FCFS outperformed AC_{Random}-FCFS in decreasing jKR by 47.06% and WPU by nearly 100%.

- AC_jAC_t-EDF outperformed AC_{Random}-EDF in decreasing jKR by 67.63% and WPU by 100%.

- AC_jAC_t-FCFS outperformed $AC_{jSlack/CR}$-FCFS in decreasing jKR by 88.89% and WPU by nearly 100%.

- AC_jAC_t-EDF outperformed $AC_{jSlack/CR}$-EDF in decreasing jKR by 82.21% and WPU by 100%.

In conclusion, our AC_jAC_t algorithm outperformed our previous FAC algorithm and the baselines admission controllers in reducing both jKR and WPU with both FCFS and EDF.

5.6.4 Profiling Killed Jobs

In this section, we are going to profile jobs that were killed by all admission control algorithms with respect to their jC values and the time (instances) at which they were decided to be removed, refer to Figure 5.11 and Figure 5.12for more insights. From Figure 5.11 and Figure 5.12, we notice:

- All algorithms followed a similar trend when the cluster becomes overloaded from the start of the simulation until around the 50^{th} hour. They start with removing smaller jobs (jC-wise) first before killing larger jobs as time progresses. This reflects on the nature of the early CFD jobs being generated and submitted into the system for scheduling and processing. Whereas later on during the simulation, larger jobs representing meshing CFD jobs are generated and submitted by end-users which are naturally larger in jC values.

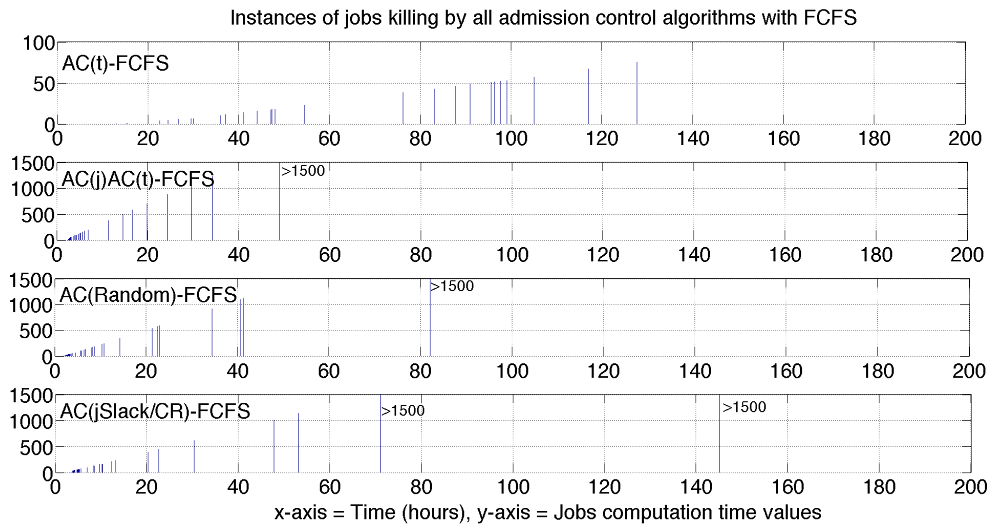

Figure 5.11: Profiling jobs killing by the admission control algorithms with FCFS with respect to killed jobs' jC values and the time of rejection.

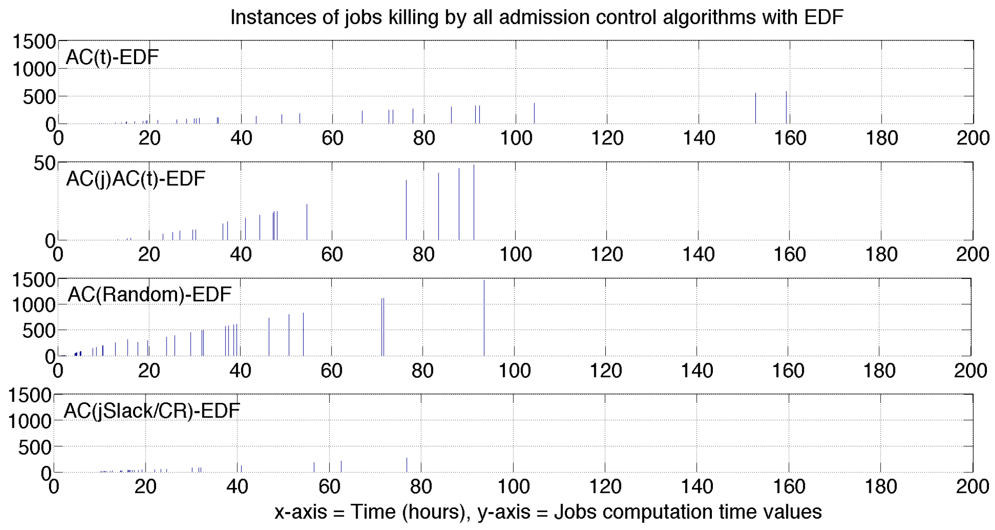

Figure 5.12: Profiling jobs killing by the admission control algorithms with EDF with respect to killed jobs' jC values and the time of rejection.

- AC_t scored one of the latest killing decisions because of its tasks-level admission mechanism. If a non-critical task becomes late, it does not necessarily imply that the whole job which contains this late task is actually late. Because of that, we see AC_t practised some late killing decisions, simulation time-wise.
- AC_jAC_t-EDF was the only algorithm that killed small jobs in comparison to its FCFS-based peer and other admission control algorithms. We already know form the previous section that AC_jAC_t-EDF scored 13.13% higher jKR than its FCFS-based peer. However, the EDF-based algorithm managed not to compromise the larger jobs in its killing decisions as the FCFS-based peer and other algorithms did.

5.6.5 Enhancing AC_{Random}

AC_{Random} shared similar performance in comparison to both of our previously and newly proposed algorithms. This can be manifested through the metrics; njS see Figure 5.7 and PU see Figure 5.8. So, we decided to increase the killing ratio of the random admission controller, named as "$AC_{Random+}$", and observe any improvements with respect to our RTS metrics as well as jKR and WPU. Figure 5.13 reviews $AC_{Random+}$'s performance with high workloads, 120%.

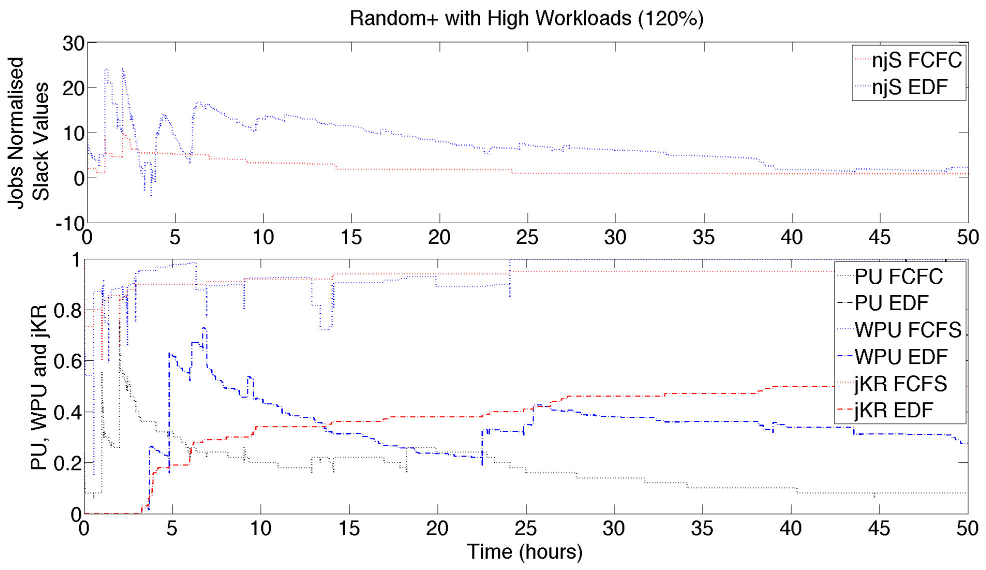

Figure 5.13: Enhanced AC_{Random} algorithm with FCFS & EDF and high workloads, 120%.

As expected, we notice an increase in njS values, in fact $AC_{Random+}$ scored positive njS values, similar to our AC_jAC_t. However, the longest $SimTime$ for $AC_{Random+}$ was 129.56 hours in comparison to its peer, AC_{Random}, which scored 246.45 hours of $SimTime$. This means that the higher the killing ratio the better the RTS performance with respect to njS, but the lesser the processing i.e. cluster PU due to the increase in killing decisions. We can notice the clear increase at both WPU & jKR, in Figure 5.13, as a consequence of killing more jobs from the system.

If we compare $AC_{Random+}$-FCFS with its EDF peer, we can see a decrease in WPU & jKR and higher njS as expected EDF would bring such advantages via its dynamic scheduling. However, neither the FCFS version of $AC_{Random+}$ nor the EDF could outperform our proposed algorithm in managing the trade-offs between our RTS objectives (maximum njS & PU) with minimal WPU & jKR altogether.

5.7 Summary

In this chapter we addressed task dependencies as an RTS constraint, and because of it the notion of jobs has emerged. Due to the notion of jobs, further RTS metrics were introduced in this chapter including jKR and WPU. We proposed a new, novel MPC-based FAC algorithm, AC_jAC_t, that follows the same admission control structure as our previous algorithm in chapter 4 where we added jobs-specific MPC and Admission Control blocks.

From the evaluations section we can see the efficiency of both of our MPC-based FAC algorithms where AC_jAC_t had the clear advantage in improving the RTS performance with respect to all of the metrics we reviewed. Whereas the other algorithms can perform good in some metrics and degrade in others. This is because the algorithm we propose in this chapter does admissions on the jobs-level. We propose AC_jAC_t as the answer to our third research question in chapter 3 which is: "*Can the proposed algorithm deal with task dependencies (inside jobs) as a scheduling constraint and the emerging RTS objectives such as reducing wasted processor utilisation (WPU)*[1]*? Can the proposed algorithm manage the trade-offs between the RTS objectives without prior knowledge to any RTS parameters? If so, can the proposed algorithm outperform baseline admission control algorithms such as random-based and jobs-completion-ratio-based etc.?*".

We also compared our newly proposed algorithm with other admission control baselines. Our proposed algorithm outperformed the baselines in all of the case-studies except in scoring the highest PU. This can be justified due to our algorithm's management of the trade-offs between the RTS objectives.

[1]We designed WPU as an RTS metric when addressing the issue of task dependencies where chapter 5 looks into it in more details.

However, it did score the longest *SimTime*.

On another note, the system administrator usually acquires additional computing processors to further improve the local HPC cluster's RTS performance with respect to the RTS objectives. In our industrial system, they acquire processors statically via annual purchases of a fixed amount of processors. We claim that our control-theoretic approach can replace the manual admission control practised by the system administrator. Thus, we shall explore how can we extend our approach into handling real-time ERP i.e. renting and buying processors during industrial projects life-time. This issue is reviewed in the next chapter.

Chapter 6

A Preliminary Investigation of Feedback Admission Control for Cloud and ERP Applications

The motivation for this chapter is to address our fourth and final research question which is concerned with supporting ERP applications while managing the trade-offs between our RTS objectives. Some industries opt for acquiring additional computing processors in order to assist their "local" cluster in meeting jobs deadlines [12]. This option is often constrained by the project's allocated budget which allows project managers to rent or buy additional computing processors when the local cluster degrades in RTS performance.

We present in this chapter novel MPC-based FAC algorithms that eliminate the need for the manual decision-making process for both admitting/rejecting jobs into the cluster and when (and how much) to rent/buy additional processors. We aim in this chapter to show that control theory is not only promising for handling the aforementioned classical real-time objectives. Additionally, it can be a productive part of the algorithm design for ERP as a Task Allocation problem was addressed in [32] using a simple feedback admission controller but without including financial constraints and incentives.

Our contributions in this chapter are:

- Embedding rent and buy algorithms in our MPC-based FAC algorithm in order to facilitate ERP applications.
- Constructing a system model for capturing the RTS dynamics between job computation time and financial values (inputs) and job normalised

slack values, processor utilisation and overall project financial credit (outputs).

- Managing the trade-offs between our RTS objectives in addition to the financial incentives and constraints.

6.1 Task Model

We used the same task model in chapter 5. We added two new notions as part of the RTS problem: *ProjectBudget* and *SystemCredit*. The idea of adding *ProjectBudget* is to allow for the automatic decision-making on rent/buy computing processors on demand in order to meet jobs deadlines and minimise the jobs killing. *ProjectBudget* is a fixed value allocated before projects start processing. We used two *ProjectBudget* values: £1000 & £500.

We have chosen these two values representing sub-budgets. We chose small amounts because the simulations we carried out in this thesis represent a sub-part of the industrial system in regards to the workload and the cluster HPC test-beds. We can consider real-life projects to be a collection of sub-projects where each project has its own allocated budget. Any left-overs from the allocated budget is considered a bonus which can be added to the *SystemCredit* towards the end of the project (simulation). The idea of adding *SystemCredit* is to reflect on jobs' individual values (jV) representing their financial incentive towards the industrial project they belong to, see Equation 6.1.

$$\begin{aligned} SystemCredit = \sum j_i V + ProjectBudgetBonus; \\ \forall i \in [JobsMetDeadlines] \end{aligned} \tag{6.1}$$

We designed jV to have a constant value throughout the job's life-cycle i.e. form generation until completion. We randomised jV as a function of the job's jC value, so we can have jobs with large jC values but a smaller jV, and vice-versa, see Equation 6.2.

$$\begin{aligned} jV = jC * Factor; \\ Factor = random(0.5 : 1.5); \end{aligned} \tag{6.2}$$

This way we can represent the financial values of jobs in our industrial system. Some jobs can have low financial values and others can have a larger incentive to process.

6.2 Scheduling Policy

This remains the same as in chapter 5.

6.3 Performance Metrics

- PU, see Equation 3.1.
- njS, see Equation 5.3.
- jKR, see Equation 5.5.
- WPU, see Equation 5.4.
- *ProjectBudget* and *SystemCredit*, see Equation 6.1.

Our RTS objectives in this chapter are:

1. Manage the trade-offs, in real-time, between maximising njS and PU.
2. Reduce jKR and WPU with the aid of rent and buy FAC algorithms.
3. Optimal use of *ProjectBudget*. Optimality here refers to minimal use of this allocated budget that would only improve meeting deadlines but without prematurely using it. Although it is intuitive to pinpoint that buying processors through *ProjectBudget* has the advantage of keeping the processors beyond the project, unlike renting. This chapter looks at the financial advantages for project managers, who are under pressure to deliver solely for the project they are currently involved with without conducting post-project and long-terms analysis on the newly acquired processors.
4. Meet a specific *SystemCredit* value which would be assigned before the beginning of the industrial project. The objective is to reach this value or near it.

6.4 System Model and Identification

We followed the same identification steps in chapter 5 for obtaining a system model to capture the correlations (system dynamics) between jC & jV (inputs as depicted from Table 5.1) and njS, PU & *SystemCredit* (outputs). The reason we included jV and *SystemCredit* into our modelling is to find out how the financial incentive is affected by the admission control decisions throughout the project life-cycle. As for the system model's accuracy, it scored 96.54% where the error plot is in Figure 6.1. The model we obtained is placed in the System Model block in Figure 6.2.

6.5 Closed-Loop Scheduling Control

We noticed in our proposed algorithm in chapter 5 that AC_jAC_t performed well due to its jobs admission controller where the tasks admission controller

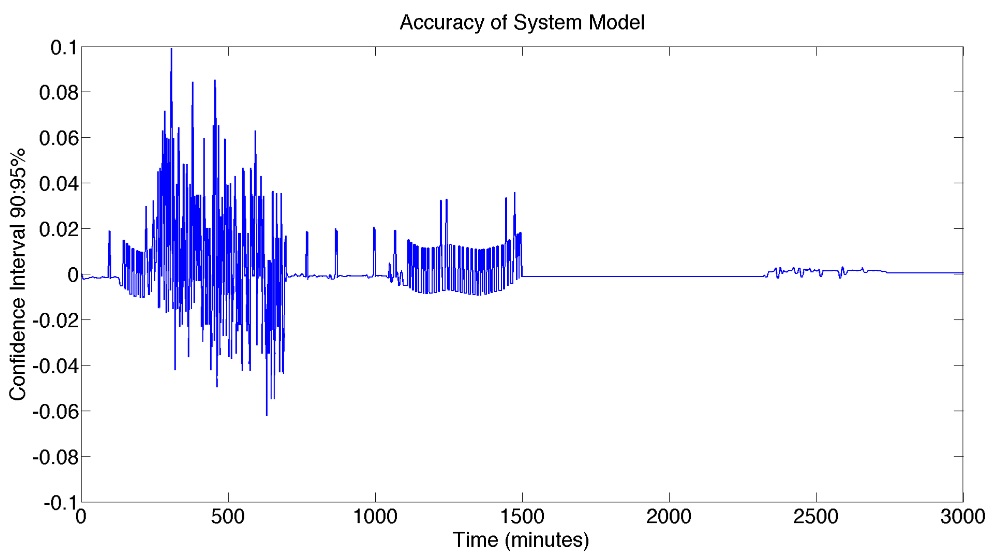

Figure 6.1: System identification model error: measured minus simulated data. The reason the x-axis is not the full 200 hours is because in system identification we allocate a small part of the measured data for the model estimation and the rest for model validation [46]..

was obsolete. Therefore, when we started adopting ERP algorithms in our AC_jAC_t algorithm, we removed the AC_t component leaving one MPC block for the jobs-level admission. Our ERP algorithms are embedded in the Admission Control block, we refer to it as VAC due to its value-based admission control logic depending on jV and $ProjectBudget$ in real-time, see Figure 6.2. In fact, if we run our current VAC algorithm without any $ProjectBudget$, it will perform the same as AC_jAC_t.

6.5.1 A Novel MPC-based VAC Algorithm for Cloud and ERP Applications

The MPC block, in Figure 6.2, receives particular system outputs (PU, njS and $SystemCredit$) which uses them for computing its control signal, $jC_{Control}$. The control signal represents the optimal jC value the local cluster can process without violating its QoS levels. We adopted two different value-based algorithms: rent-on-demand (RoD) and purchase-on-demand (PoD). Each one of them follows a different admission control logic which we used in our evaluations in order to explore the (dis-)advantages each algorithm can bring towards our real-time objectives. Both algorithms have the same structure as depicted in Figure 6.2. We will collectively refer to both algorithms as the Value-based Admission Control block (VAC).

As for controllers' stability, we tested our VAC controller design inside MPC toolbox in Matlab environment where, similarly, to our previous controllers

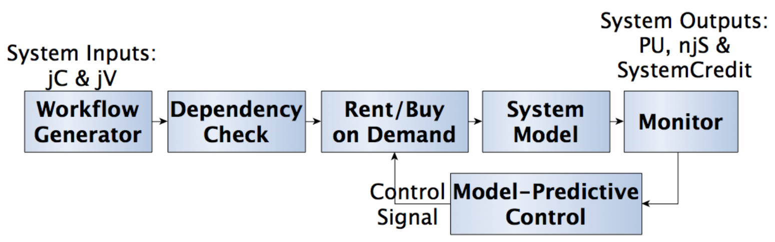

Figure 6.2: A block diagram of our novel value-based admission control (VAC) algorithm.

in chapter 5, it passed all tests with a warning in regards to the controller's steady-state gains. The warning suggests sacrificing one output's weight values to zero in order to reduce possible errors in other outputs. However, we opted for full weight values for all system outputs because they are all equally important to our RTS optimisation problem.

Before we address each algorithm, it is important to define two metrics:

- RoD Cloud Price = £0.05 per core per hour on the cloud [51]. We do not include the time overheads for jobs to be sent to cloud and queue there. We assume once an admission decision is made to transfer a job to the cloud, it will receive instant processing capability.
- PoD HPC Rack Price = £250 per a 15-processor HPC rack [51]. We do not include the delivery time overheads of HPC racks as part of our optimisation control at this stage of our research. We assume once an admission decision is made to acquire HPC racks, the local cluster will receive it instantly.

Rent-on-Demand Admission Control Logic

```
if Job Computation Time <= MPC signal
    send job to the local cluster;
else
    (if Job Value >= RoD Price) &&
    (ProjectBudget >= RoD Price)
        rent on the cloud;
    else
        kill job;
    end
end
```

As long as newly arrived job's jC values are within the control signal value, $jC_{Control}$, the controller predicts for continuing meeting the real-time objectives using the local cluster. Otherwise, it is believed this job will not meet its deadline and the admission logic here rejects processing the job locally. At this stage, the logic weighs the financial incentive against the price for renting additional computing processors on the cloud. There can be a rejection at the cloud stage in two scenarios:

- jV isn't enough to pay-back the cloud price, regardless whether we have enough budget *now* or not.
- We do not have enough *ProjectBudget now* for running the job on the cloud, regardless whether the job is financially valuable or not.

Purchase-on-Demand Admission Control Logic

```
if Job Computation Time <= MPC signal
    send job to the local cluster;
else
    if ProjectBudget >= PoD Price
        purchase new rack;
    else
        kill job;
    end
end
```

Similarly to RoD, as long as the newly arrived job's jC value is less than or equal to $jC_{Control}$, the job runs on the local cluster. Otherwise, the PoD logic looks into buying additional computing processors. The purchase will occur for a 15-processor rack, instead of renting individual processors on the cloud as in RoD. Once the transaction occurs after checking whether our *ProjectBudget* allows to authenticate the purchase transaction, the newly acquired rack becomes a physical part of the local cluster. This way, the opportunity for processing more jobs locally increases which potentially reduces the need for acquiring additional resources.

6.6 Evaluations

The workflows depicted from Table 5.1 using Burkimsher's generator were tested in a 50-processor HPC cluster. Each workflow contains jobs being generated at least three times, we ran each case-study 10 times and we averaged the simulation results in the metrics plots. The allocated *ProjectBudget* values are £1000 and £500 representing high and average budgets for sub-projects in our industrial organisation. One-way analysis of variance (ANOVA) tests were carried-out to statistically demonstrate the difference between the results obtained from experimental case-studies. We will refer to each case-study by the admission algorithm (OL, AC_jAC_t, RoD and PoD) concatenated with the scheduling policy experimented with. For instance, RoD-EDF refers to our rent-on-demand algorithm with EDF.

For each RTS metric, we will review:

- RTS analysis: the performance of EDF against FCFS per algorithm.
- Admission Control analysis: the performance of all systems against each other.

Please note that full numeric results for all workload types (low, full and high) are tabulated in the Appendix in Table C.1. The high workload cases are reviewed in this section due to their high RTS dynamics that will reflect more on the different performances of the algorithms. We will not include the notion of simulation time ($SimTime$) as we did in the previous chapter. This is because our VAC algorithms can opt for acquiring additional computing processors to achieve the RTS objectives. Thus, a smaller $SimTime$ for RoD does not necessarily reflect performance degradation in keeping the local cluster busy throughout the simulation. Instead, it reflects on the local cluster's incapability to continue achieving the RTS objectives, hence cloud decisions were made.

6.6.1 Job Normalised Slack Values

For this metric, we expect:

- Similarly in the previous chapter, the OL algorithms to score the lowest and negative njS values.
- The closed-loop algorithms (AC_jAC_t, RoD and PoD) to score ≥ 0 njS values.
- Higher-budgeted VAC algorithms to score higher njS values than their lower peers because of the capability of acquiring more additional computing processors.

For RTS analysis, we expect EDF to outperform FCFS in scoring higher njS values, refer to Figure 6.3 for more insights and Table 6.1 for statistical findings:

- In the OL algorithm, EDF scored 163.06% more than FCFS. This is a good result for EDF but it still scored negative njS values beyond the QoS level, -1.
- In AC_jAC_t, EDF scored 4.37% more than FCFS.
- In RoD£1000, EDF scored 82.40% more than FCFS.
- In RoD£500, EDF scored 300.25% more than FCFS.
- In PoD£1000, EDF scored 223.45% more than FCFS.

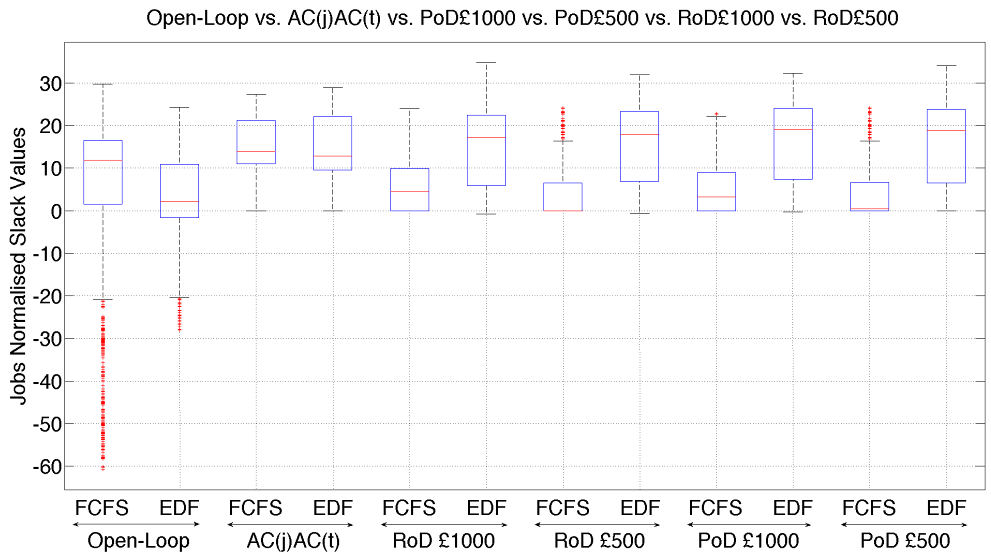

Figure 6.3: Jobs normalised slack values box-plots in high workloads, 120%.

- In PoD£500, EDF scored 281.57% more than FCFS.

Overall, we notice that EDF in the VAC algorithms were more effective in comparison to EDF in AC_jAC_t in increasing njS values. This can be explained by the fact that with VAC we can outsource jobs to additional computing processors (cloud or new HPC racks) supporting the local cluster in meeting deadlines. However with AC_jAC_t, there is only the local cluster to process all arriving jobs which made the cluster busier than with the VAC algorithms. With overloaded cluster the scheduling policy becomes less effective. Yet, all closed-loop algorithms scored ≥ 0 njS values as expected.

For admission control analysis, refer to Figure 6.3 for more insights and Table 6.2 for statistical findings:

- AC_jAC_t-FCFS scored 336.94% more than OL-FCFS.
- AC_jAC_t-EDF scored 73.37% more than OL-EDF.
- RoD£500-FCFS scored 70.48% less than AC_jAC_t-FCFS. This counter-intuitive result can be explained by the high jKR incurred by AC_jAC_t than RoD, see subsection 6.6.4 for more details. The higher the jKR the higher the njS values which is why AC_jAC_t scored, in average, higher njS values. Yet, RoD also scored ≥ 0 njS values.
- RoD£500-EDF scored 13.20% more than AC_jAC_t-EDF.
- RoD£1000-FCFS scored 50.12% more than RoD£500-FCFS.
- RoD£1000-EDF scored 5.92% more than RoD£500-EDF.

Case-Study (FCFS vs. EDF)	***p*-value**
OL	$9.26e^{-143}$
AC_jAC_t	$0.05e^{-2}$
RoD£1000	0.00
RoD£500	0.00
PoD£1000	0.00
PoD£500	0.00

Table 6.1: One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of njS.

Case-Study	***p*-value**
OL-FCFS vs. AC_jAC_t*-FCFS*	$5.1e^{-170}$
OL-EDF vs. AC_jAC_t*-EDF*	0.00
AC_jAC_t*-FCFS vs. RoD£500-FCFS*	0.00
AC_jAC_t*-EDF vs. RoD£500-EDF*	0.00
RoD£500-FCFS vs. RoD£1000-FCFS	$1.81e^{-107}$
RoD£500-EDF vs. RoD£1000-EDF	0.01*
PoD£500-FCFS vs. PoD£1000-FCFS	$2.12e^{-23}$
PoD£500-EDF vs. PoD£1000-EDF	***0.47***
RoD£500-FCFS vs. PoD£500-FCFS	$4.05e^{-7}$
RoD£500-EDF vs. PoD£500-EDF	$0.03e^{-2}$
RoD£1000-FCFS vs. PoD£1000-FCFS	$9.33e^{-13}$
RoD£1000-EDF vs. PoD£1000-EDF	$5.26e^{-12}$

Table 6.2: One-way ANOVA test for the admission control experimental data of njS. Mark (*) indicates that the difference between the data-sets is almost insignificant.

- PoD£1000-FCFS scored 18.89% more than PoD£500-FCFS.
- PoD£1000-EDF scored 0.79% more than PoD£500-EDF.
- RoD£500-FCFS scored 6.91% less than PoD£500-FCFS.
- RoD£500-EDF scored 2.14% less than PoD£500-EDF.
- RoD£1000-FCFS scored 17.83% less than PoD£1000-FCFS.
- RoD£1000-EDF scored 2.88% more than PoD£1000-EDF.

In most cases, the larger the *ProjectBudget* the larger the njS values because with a higher budget the organisation can afford a larger computing capacity which can increase the opportunity for processing more jobs on time if not early. We also notice that both RoD and PoD show similar values. The reason they are different and not exactly the same is because PoD adds the newly purchased HPC racks into the local cluster. So, the local cluster size increases which makes the MPC's prediction and optimisation decisions differ from RoD's MPC because now we are dealing with two different local cluster sizes

With RoD, the transferred job to the cloud is assumed to get instant access to the required number of processors which will enable the job to meet its deadline quicker. As for the PoD case, when it decides to purchase a new HPC rack, jobs still need to queue to access those processors and hence they might take longer to start processing than in the RoD case. However, both algorithms acquire their additional computing processors knowing in advance that jobs will meet their deadlines via MPC's prediction and optimisation algorithms.

In conclusion to this metric:

- The VAC algorithms and in particular with EDF outperformed both OL and AC_jAC_t in increasing njS values.
- The higher the $ProjectBudget$ the higher the njS values in all VAC case-studies.

6.6.2 Processor Utilisation

For this metric, we expect:

- The OL algorithms to score the highest PU values because they do not practice in jobs killing or outsourcing.
- RoD to score higher PU than PoD, because PoD can increase the local HPC's cluster size when purchasing new racks resulting in needing some time for the cluster to be fully utilised.

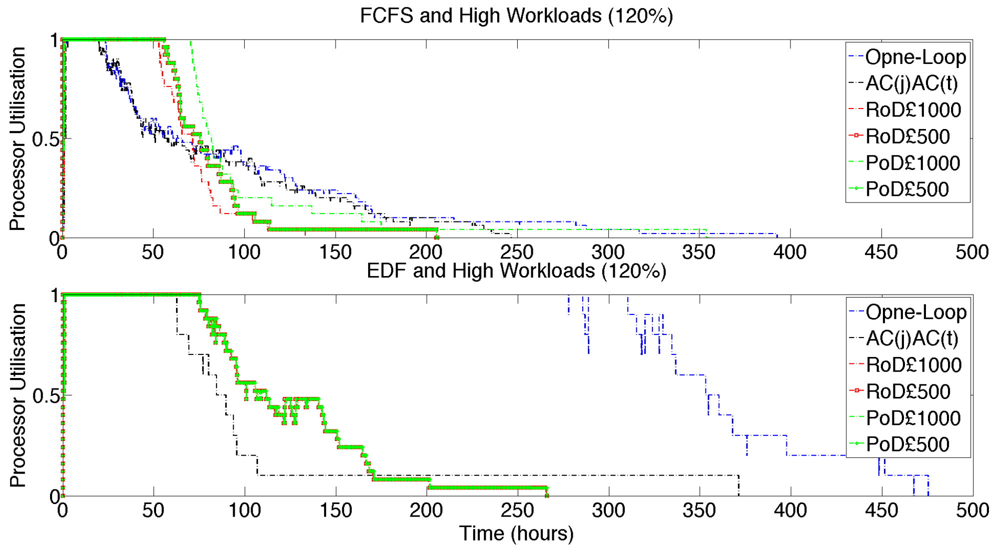

Figure 6.4: Processor utilisation time-plots in high workloads, 120%.

For RTS analysis, we expect EDF to outperform FCFS in scoring higher PU values, refer to Figure 6.4 for more insights and Table 6.3 for statistical findings:

Case-Study (FCFS vs. EDF)	***p*-value**
OL	0.00
AC_jAC_t	0.00
RoD£1000	$5.11e^{-16}$
RoD£500	0.01*
PoD£1000	$1.27e^{-14}$
PoD£500	***0.09***

Table 6.3: One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of *PU*. Mark (*) indicates that the difference between the data-sets is almost insignificant.

Case-Study	***p*-value**
OL-FCFS vs. AC_jAC_t*-FCFS*	$3.1e^{-29}$
OL-EDF vs. AC_jAC_t*-EDF*	$2.03e^{-11}$
AC_jAC_t*-FCFS vs. RoD£500-FCFS*	$1.9e^{-279}$
AC_jAC_t*-EDF vs. RoD£500-EDF*	$3.46e^{50-}$
RoD£500-FCFS vs. RoD£1000-FCFS	$6.18e^{-12}$
RoD£500-EDF vs. RoD£1000-EDF	***0.94***
PoD£500-FCFS vs. PoD£1000-FCFS	$1.76e^{-32}$
PoD£500-EDF vs. PoD£1000-EDF	***0.27***
RoD£500-FCFS vs. PoD£500-FCFS	***0.09***
RoD£500-EDF vs. PoD£500-EDF	***0.31***
RoD£1000-FCFS vs. PoD£1000-FCFS	$0.14e^{-2}$
RoD£1000-EDF vs. PoD£1000-EDF	***1.00***

Table 6.4: One-way ANOVA test for the admission control experimental data of *PU*.

- In the OL algorithm, EDF scored 16.93% more than FCFS.
- In AC_jAC_t, EDF scored 20.43% more than FCFS.
- In RoD£1000, EDF scored 1.57% more than FCFS.
- In RoD£500, EDF scored 1.52% less than FCFS.
- In PoD£1000, EDF scored 4.21% more than FCFS.
- In PoD£500, EDF scored 0.57% less than FCFS.

For the OL and AC_jAC_t algorithms, we notice that EDF was better than FCFS in increasing *PU* because both algorithms do not have access to additional computing processors. Once we enabled acquiring additional resources we noticed a decrease in the local cluster's *PU* due to renting from the cloud in the RoD case.

For admission control analysis, refer to Figure 6.4 for more insights and Table 6.4 for statistical findings:

- AC_jAC_t-FCFS scored 3.01% less than OL-FCFS.

- AC_jAC_t-EDF scored 0.49% more than OL-EDF.
- RoD£500-FCFS scored 15.06% more than AC_jAC_t-FCFS.
- RoD£500-EDF scored 6.89% less than AC_jAC_t-EDF.
- RoD£1000-FCFS scored 5.77% less than RoD£500-FCFS.
- RoD£1000-EDF scored 0.05% less than RoD£500-EDF.
- RoD£1000-FCFS scored 0.01% more than PoD£1000-FCFS.
- RoD£1000-EDF scored 0% than PoD£1000-EDF.
- PoD£1000-FCFS scored 6.37% less than PoD£500-FCFS.
- PoD£1000-EDF scored 1.59% less than PoD£500-EDF.

A reduction in PU from RoD to PoD does not necessarily refer to performance degradation. Instead, it means the algorithm did utilise additional computing processors, from the cloud, which led to lesser utilisation of the local cluster.

In conclusion to this metric:

- RoD experienced low PU values due to outsourcing jobs to the cloud.
- PoD experienced low PU values due to increasing the local HPC cluster's size.

6.6.3 Project Budget

For this metric, we expect:

- Lower-budgeted VAC algorithms to use their full *ProjectBudget* value in comparison to the higher peers leading to another expectation of higher-budgeted VAC algorithm to accumulate some *ProjectBudget* bonuses.
- A higher *ProjectBudget* bonus with EDF-based VAC algorithms in comparison to their FCFS peers, because EDF's better performance in meeting deadlines will reduce the need for acquiring additional computing processors. Hence, the higher *ProjectBudget* bonus towards the end of the simulation (industrial project life-time).
- Amongst the VAC algorithms, we expect RoD to use less of its allocated *ProejctBudget* in comparison to PoD because RoD does cloud transactions for a specific number of processors per job whereas PoD does HPC transactions for HPC racks (not processors).

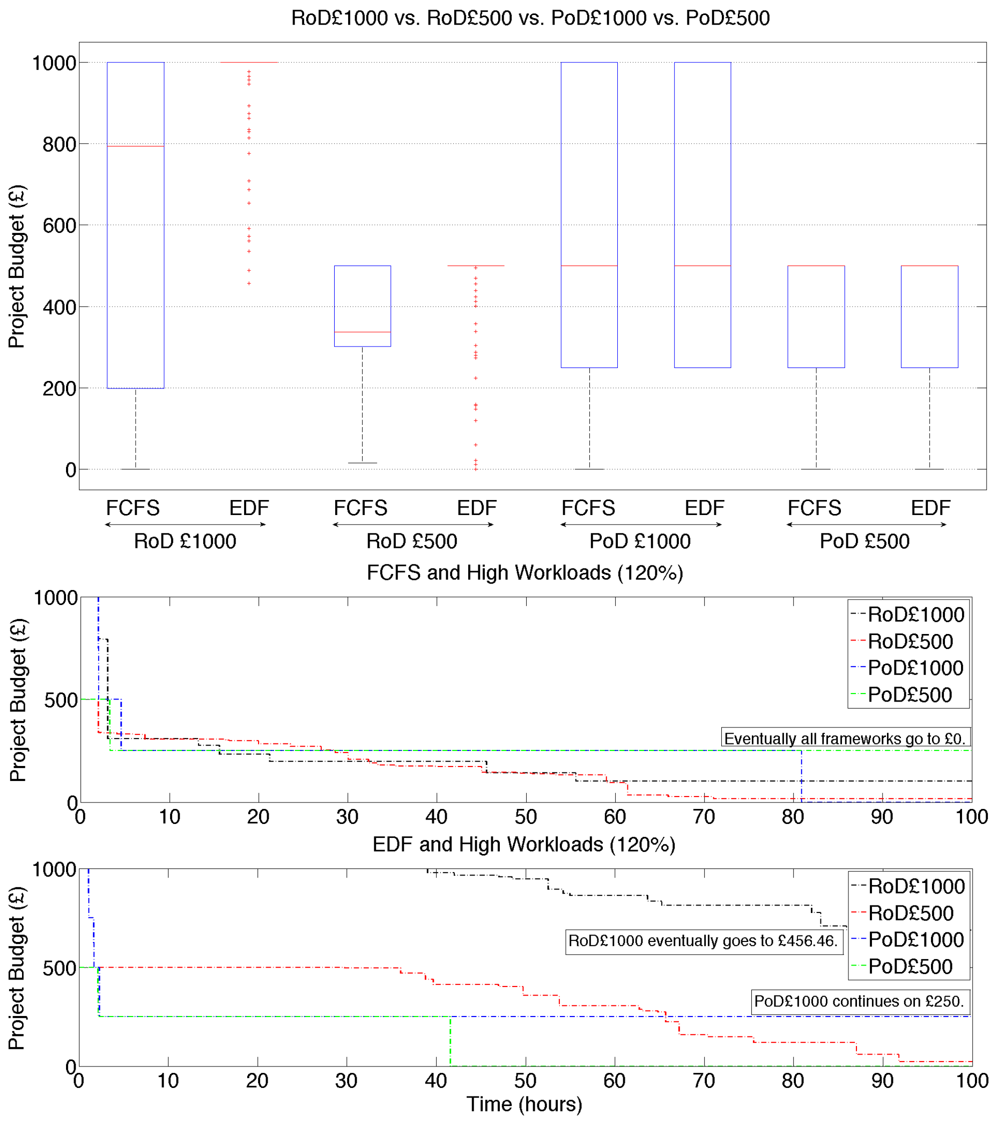

Figure 6.5: Project budget plots in high workloads, 120%. We zoomed-in to show the RTS dynamics during the cluster overload period.

We notice from Figure 6.5 that the savings from the RoD-EDF algorithm were £456.46 while PoD-EDF scored £250 i.e. £206.46 less. Using parts or all of the *ProjectBudget* reflects the fact that the VAC algorithms have utilised it for improving the njS values which will lead to lesser jKR if not eliminate it which is desired.

The difference between RoD and PoD's savings can be explained by the fact that PoD does individual purchases of HPC racks, each is worth £250. The PoD-EDF algorithm in our experiments made three purchases for racks: first, £1000 - £250 = £750; second, £750 - £250 = £500; and third, £500 - £250 = £250. PoD-FCFS made an additional fourth purchase due to FCFS poor scheduling performance in meeting deadlines resulting in needing more additional processors which has emptied the allocated *ProjectBudget*. Whereas RoD rents a specific number of processors per job on the cloud and the price varies based on the number and duration of the rented processors.

In conclusion to this metric:

- The higher the *ProjectBudget* the higher the savings for the same algorithm.
- In this sense, we find that RoD is more dynamic than PoD. This is because the latter does purchases of racks whereas RoD rents exactly what is needed for particular jobs at a time.

6.6.4 Jobs Killing Ratio and Wasted Processor Utilisation

For these metrics, we expect:

- The VAC algorithms to reduce jKR & WPU more than AC_jAC_t. This is because of their ability in acquiring additional processors which reduces the need for killing jobs and hence reduced WPU as well.
- Higher-budgeted VAC algorithms to score the least jKR & WPU if not eliminate them.

From Figure 6.6 and Figure 6.7, we can generally notice that our VAC algorithms showed lesser jKR than AC_jAC_t. All VAC and FAC algorithms scored relatively small WPU values <1%. This reflects the good performance in killing jobs early enough without causing WPU. All VAC algorithms with £1000 *ProjectBudget* have shown 0 jKR indicating the benefit in having a financial aid for acquiring additional computing processors where it eliminated the killing of jobs from the system. Whereas with £500 *ProjectBudget*, our VAC algorithms have killed a number of jobs. This is because the small budget was not enough to process all of the arrived jobs locally and have them outsourced to the cloud or extra HPC racks.

Our VAC algorithms with £1000 *ProjectBudget* have caused 0% WPU, which is a 100% improvement from AC_jAC_t. The VAC algorithms with £500 incurred WPU due to their jKR because of the lower *ProjectBudget*.

For RTS analysis, refer to Figure 6.6 and Figure 6.7 for more insights and to Table 6.5 for statistical findings:

- In AC_jAC_t, EDF increased jKR by 13.13% while decreased WPU by 100%. The EDF-based algorithm incurred an increase in its jKR but scored 0 WPU in comparison to the FCFS-based peer.

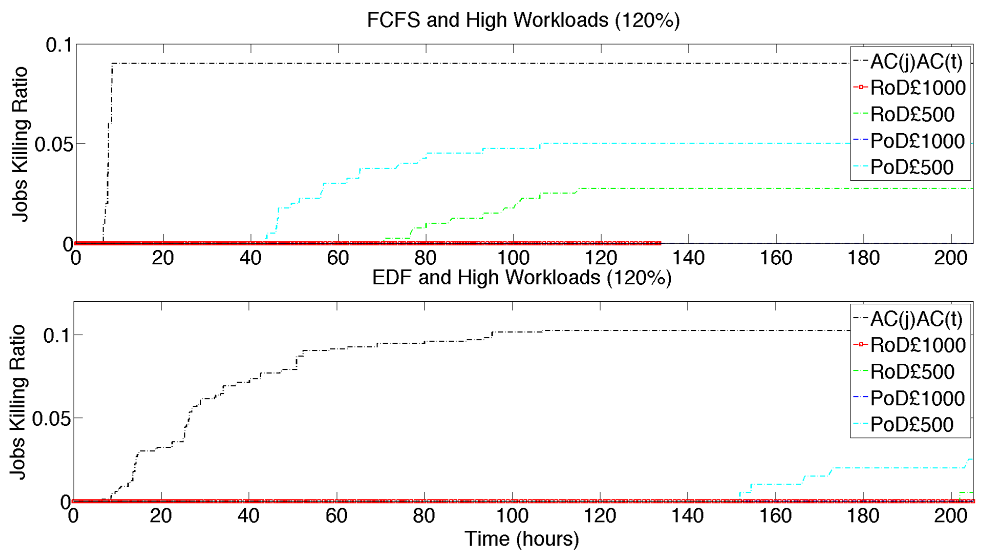

Figure 6.6: Jobs killing ratio time-plots in high workloads, 120%. We zoomed-in to show the RTS dynamics during the cluster overload period.

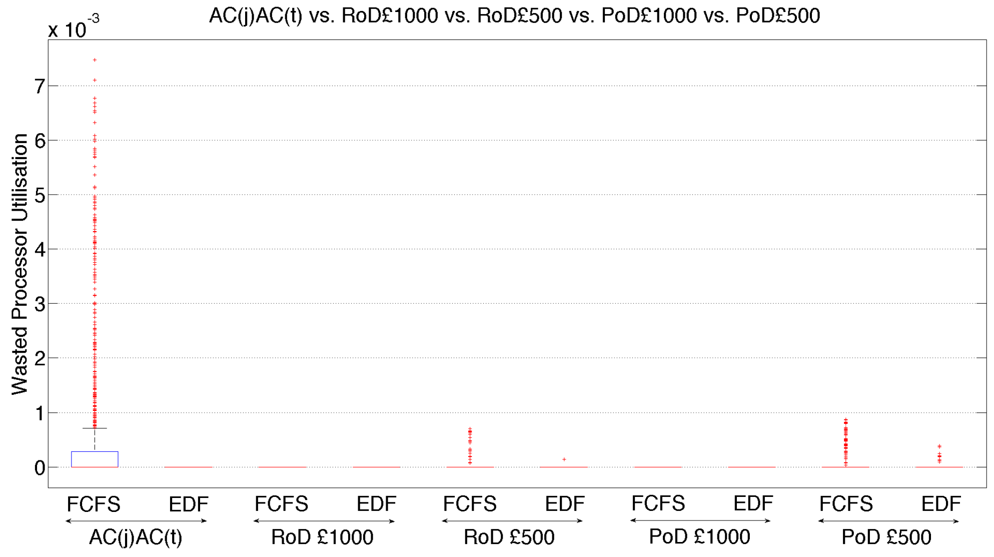

Figure 6.7: Wasted processor utilisation box-plots in high workloads, 120%.

Case-Study	jKR ***p*-value**	WPU ***p*-value**
AC_jAC_t	$3.03e^{-3}$	0.00
RoD£1000	NaN**	NaN**
RoD£500	$1.07e^{-29}$	$1.09e^{-29}$
PoD£1000	NaN**	NaN**
PoD£500	$4.19e^{-32}$	$2.34e^{-32}$

Table 6.5: One-way ANOVA test for the RTS (FCFS vs. EDF) experimental data of jKR and WPU. Mark (**) indicates that the data-sets are equal; when we ran the ANOVA tests in both Matlab and R programming environments, we obtained p-value = NaN.

Case-Study	jKR ***p*-value**	WPU ***p*-value**
AC_jAC_t*-FCFS vs. RoD£500-FCFS*	0.00	$1.31e^{-311}$
AC_jAC_t*-EDF vs. RoD£500-EDF*	$3.4e^{-287}$	$2.59e^{-19}$
RoD£500-FCFS vs. RoD£1000-FCFS	$2e^{-67}$	$1.03e^{-67}$
RoD£500-EDF vs. RoD£1000-EDF	$4.94e^{-21}$	2.95^{-21}
PoD£500-FCFS vs. PoD£1000-FCFS	$1.22e^{-100}$	$1.75e^{-101}$
PoD£500-EDF vs. PoD£1000-EDF	$1.44e^{-39}$	$8.05e^{-40}$
RoD£500-FCFS vs. PoD£500-FCFS	$6.49e^{-9}$	$2.42e^{-8}$
RoD£500-EDF vs. PoD£500-EDF	$1.02e^{-7}$	$1.44e^{-7}$
RoD£1000-FCFS vs. PoD£1000-FCFS	NaN**	NaN**
RoD£1000-EDF vs. PoD£1000-EDF	NaN**	NaN**

Table 6.6: One-way ANOVA test for the admission control experimental data of jKR and WPU. Mark (**) indicates that the data-sets are equal; when we ran the ANOVA tests in both Matlab and R programming environments, we obtained p-value = NaN.

- In RoD£500, EDF scored 81.81% less jKR and 85.71% less WPU than FCFS.
- In PoD£500, EDF scored 50% less jKR and 50% less WPU than FCFS.

For admission control analysis, refer to Figure 6.6 and Figure 6.7 for more insights and to Table 6.6 for statistical findings:

- RoD£500-FCFS scored 69.44% less jKR and 165.85% less WPU than AC_jAC_t-FCFS.
- RoD£500-FCFS scored 45% less jKR and 12.5% less WPU than PoD£500-FCFS.
- RoD£500-EDF scored 95.17% less jKR and 100% more WPU than AC_jAC_t-EDF. Despite the large difference in WPU, both algorithms scored 0.01% and 0%, respectively, refer to Table C.1 for more details.
- RoD£500-EDF scored 80% less jKR and 75% less WPU than PoD£500-EDF.

In conclusion to this metric:

- Higher-budgeted VAC algorithms eliminated jKR (and hence WPU).
- Lower-budgeted peers incurred some jKR and WPU but significantly less than AC_jAC_t.

6.6.5 System Credit

For this metric, we expect:

- The OL algorithms to score the lowest *SystemCredit* due to their poor performance in meeting jobs deadlines. Although OL algorithms do not kill any jobs, but if a job misses its deadline, the value of that job will not be added to the *SystemCredit* basket.
- AC_jAC_t to score the second lowest *SystemCredit*. Although AC_jAC_t scored ≥ 0 njS values, but because of its higher jKR in comparison to the VAC algorithms, AC_jAC_t did not score as high *SystemCredit* as the VAC peers.
- Amongst the VAC algorithms, we expect RoD to score a higher *System–Credit* due to its higher savings in using *ProjectBudget*, refer to subsection 6.6.3 for more details.

We can notice from Figure 6.8 that the OL algorithms have scored the lowest *SystemCredit* due to their deadline misses. AC_jAC_t comes at second in scoring higher *SystemCredit*. Meanwhile, OL-EDF and AC_jAC_t-EDF have scored higher *SystemCredit* than their FCFS-based peers due to EDF's dynamic scheduling policy enabling more jobs to meet their deadlines, hence the higher credits earned. From here we can see that jKR and *SystemCredit* are inversely proportional, the higher the jKR the lesser jobs to process hence lesser jV earned reflected on the *SystemCredit* metric.

As for the VAC algorithms, we can notice that RoD have scored higher *SystemCredit* than its PoD peer, especially when experimenting with the £1000 *ProjectBudget* which has managed to earn savings from not using-up all of the allocated budget. Besides, being able to acquire additional computing processors and due to EDF's good performance in meeting deadlines, there was no need found to kill any jobs as reflected by the jKR metric in the previous sub-section.

For RTS analysis, refer to Figure 6.8 for more insights:

- In the OL algorithm, EDF scored £924.73 more than FCFS.

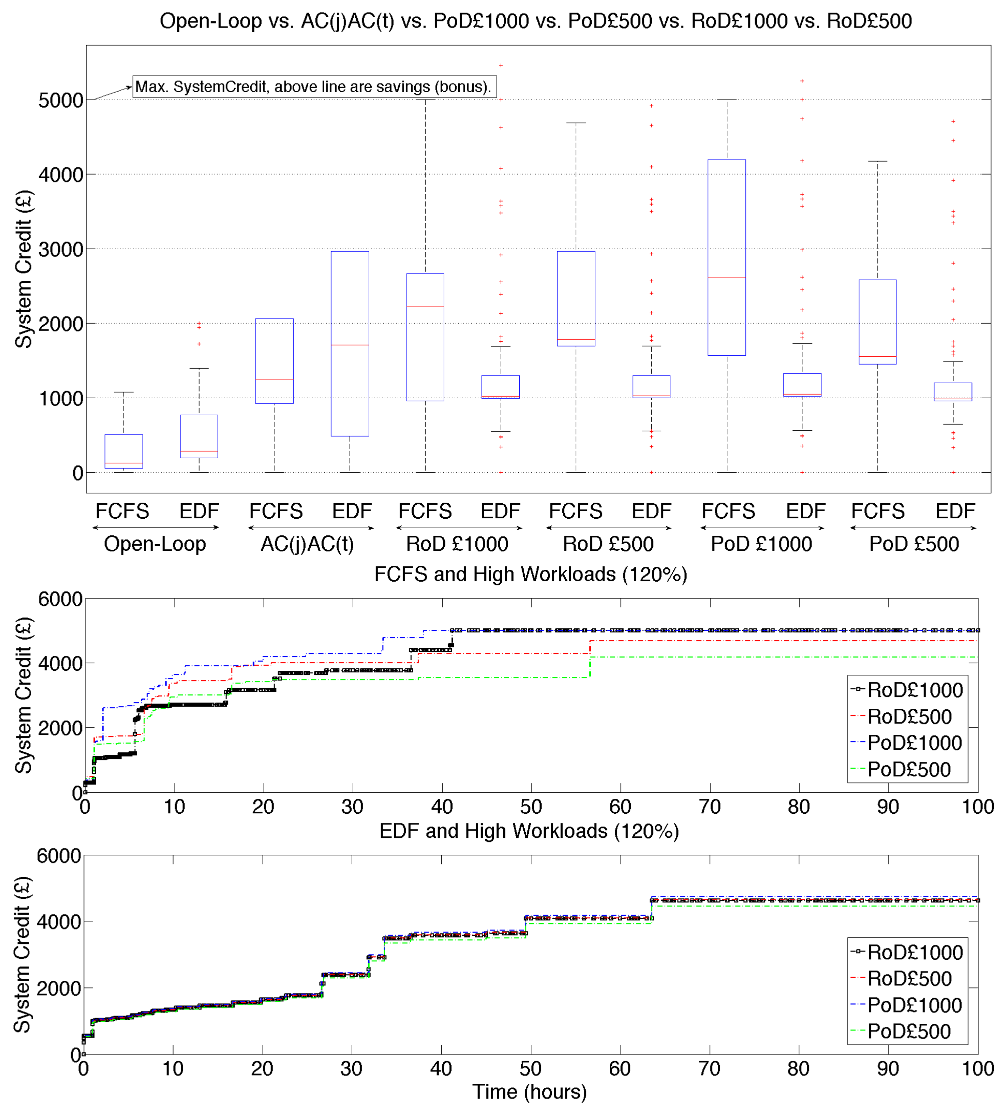

Figure 6.8: System credit plots in high workloads, 120%. We zoomed-in to show the RTS dynamics during the cluster overload period.

- In AC_jAC_t, EDF scored £904.07 more than FCFS.
- In RoD£1000, EDF scored £456.46 more than FCFS.
- In RoD£500, EDF scored £231.27 more than FCFS.
- In PoD£1000, EDF scored £250 more than FCFS.
- In PoD£500, EDF scored £529.54 more than FCFS.

There is a consistent pattern for EDF in increasing the $SystemCredit$ more than the FCFS peers across all algorithms. This can motivate implementing EDF in any given algorithm as we saw its performance in the previously reviewed RTS metrics. As for the VAC algorithms, the reason the EDF-based algorithms scored higher $SystemCredit$ than their FCFS peer in some cases is due to the lesser jKR EDF incurred, which is desired and also expected. Therefore, we can confirm our inversely proportional relationship between jKR and $SystemCredit$.

For admission control analysis, refer to Figure 6.8 for more insights:

- AC_jAC_t-FCFS scored £985.04 more than OL-FCFS.
- AC_jAC_t-EDF scored £964.38 more than OL-EDF.
- RoD£500-FCFS scored £2625.29 more than AC_jAC_t-FCFS.
- RoD£500-EDF scored £1952.49 more than AC_jAC_t-EDF.
- RoD£1000-FCFS scored £314.4 more than RoD£500-FCFS.
- RoD£1000-EDF scored £539.59 more than RoD£500-EDF.
- RoD£1000-FCFS and PoD£1000-FCFS both scored the same value of £5000 which is the maximum $SystemCredit$ and this is due to 0 jKR they both incurred across all the workloads.
- RoD£1000-EDF scored £206.46 more than PoD£1000-EDF.
- PoD£1000-FCFS scored £826.06 more than PoD£500-FCFS.
- PoD£1000-EDF scored £546.52 more than PoD£500-EDF.

In conclusion to this metric, we propose RoD-EDF for scoring higher $SystemCredit$ where, of course, the higher the $ProjectBudget$ the higher the $SystemCredit$.

6.6.6 Trade-off Analysis

In this section, we explore the trade-off when we increase and decrease the RoD cloud price from the original value we experimented with, £0.05 per core per hour. We chose £0.1 and £0.01 for testing if RoD and PoD break even or have close amount of *ProjectBudget* savings. The motivation for this section is to review from the project manager's point of view what algorithm to opt for given the RoD and PoD price lists from typical cloud and HPC service providers. The intention here is not to provide exact values per se. Instead, to steer the project manager's attention into what potential trade-offs they can deal with if they opt for one VAC algorithm to another, subject to varying financial incentives (jV) and constraints ($ProjectBudget$).

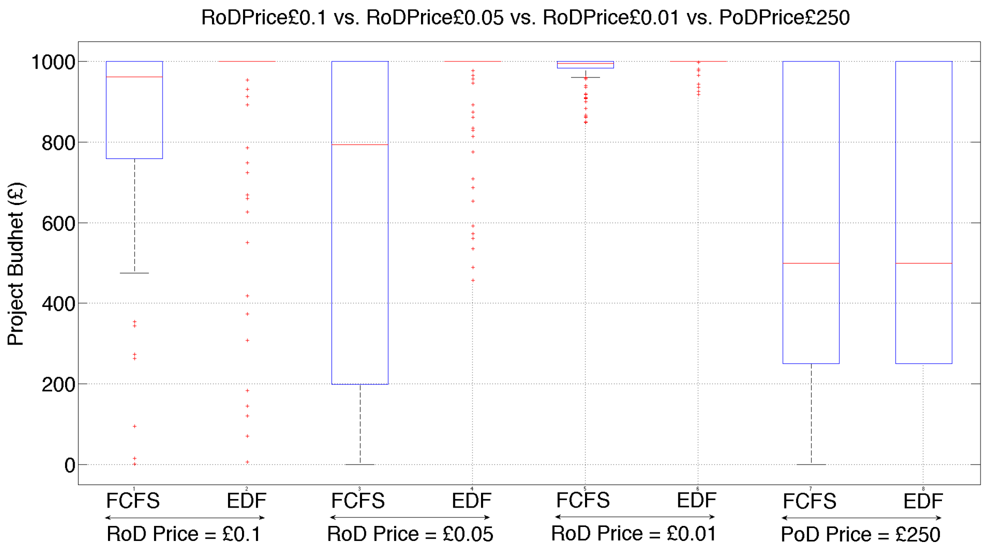

Figure 6.9: Trade-off analysis between multiple RoD rent prices and PoD in high workloads, 120%.

From Figure 6.9 we can notice that with a more expensive RoD cloud price of £0.1 per core per hour, the admission controller ends up using the entire allocated budget with both FCFS and EDF policies. This means, with a higher cloud price, savings are reduced which in our particular case here resulted to zero savings. When we tested with a lesser cloud price we were able to measure how much varying it can affect the savings, overall. We noticed a great difference in savings for both FCFS and EDF with £848.43 and £461.91, respectively. Figure 6.9 provides a general trend for savings order presented in Equation 6.3.

$$RoD£0.1 < PoD£250 < RoD£0.05 < RoD£0.01 \tag{6.3}$$

Given the industrial system's workload and processing characteristics, our Equation 6.3 can provide an intuition towards the most economic algorithm to adopt if the project manager is given various cloud offers from various service providers. RoD £0.10 is the lowest economic option and RoD £0.01 is as expected to be the best option.

6.7 Summary

Our previous chapter addressed the dependencies between soft real-time tasks where the notion of jobs, as a result, emerged. Our proposed algorithm in chapter 5 showed promising performance. However, we wanted to extend the RTS problem to address the issue of acquiring additional computing processors during projects life-times. In this chapter, we considered one of the realistic constraints in industrial systems, *ProjectBudget*, where scheduling of jobs can be done with respect to individual jobs' jV values. We have proposed two novel VAC algorithms: RoD and PoD. As a result, we have added two scheduling metrics to our RTS objectives: *ProjectBudget* and *SystemCredit*, where any left-overs from *ProjectBudget* was added to the *SystemCredit* basket as a bonus towards the end of the simulation (project life-time).

In summary, for the njS metric, all our control-theoretic algorithms (FAC and VAC) have scored positive values unlike the OL peers. For the PU metric, we noticed that the EDF-based algorithms scored higher PU values than their FCFS-based peers, which is expected for EDF to contribute to higher utilisation. We can also notice that PoD-FCFS has shown lower PU values than RoD-FCFS. This is because PoD joins the newly purchased racks to its local cluster where it would take some time for it to be fully utilised. However, RoD only rents from the cloud hence its cluster size remains fixed.

We can conclude that our RoD-based algorithms have performed better than PoD in saving more on the *ProjectBudget* and further reduce jKR. RoD with EDF is our proposed algorithm for handling task dependencies influenced by job's financial incentives and we propose it as the answer to our fourth and final research question in chapter 3 which is:
"*Can the algorithm support real-time rent and buy decisions for acquiring additional computing processors to support the local cluster in managing the trade-offs between the RTS objectives and increase the overall SystemCredit?*".

As for the trade-off analysis, we were able to generate a general trend, see Equation 6.3, relating the performance of our VAC algorithms where we varied the RoD cloud price in comparison to the original price we experimented with and in comparison to PoD. The cheaper the RoD cloud price the higher the savings at the end of the project life-cycle. Additionally, the PoD algorithm scored higher savings than RoD £0.10 but lower than £0.05 and £0.01.

Chapter 7

Conclusions and Further Work

Throughout this thesis we have developed novel admission control algorithms in addressing our three research objectives which are:

- To propose an intelligent admission controller that predicts the effects of arriving jobs on the RTS objectives via a model-predictive algorithm without prior knowledge of jobs' RTS parameters.
- To propose an admission controller that computes an admission decision for arriving jobs (unknown apriori) while respecting their task dependencies in minimising such wasted processor utilisation while managing the trade-offs between the RTS objectives.
- To propose an admission controller that can evaluate, in real-time, whether it is financially plausible to acquire additional computing processors to support the local cluster in processing more jobs while handling task dependencies and managing the trade-offs between the RTS objectives.

Our control-theoretic algorithm is best described as an MPC-based feedback admission controller for supporting the RTS system in managing the trade-offs between the RTS objectives. Our first contribution addressed the first and second research questions in chapter 3 which are:

- Can we construct a dynamic mathematical model which captures the (non-linear) behaviour between task-sets computation time values (system input) and their slack values & processor utilisation (system outputs). If so, how can we validate its accuracy in order for it to be implemented as part of the proposed algorithm?
- Once we validate our system model, can we design an intelligent admission control algorithm for managing the trade-offs between the two RTS objectives without prior knowledge of jobs RTS parameters?

We claim this thesis has addressed those question by:

- Modelling our industrial WMS using system identification techniques in order to comply with our admission controller design.
- Designing a novel MPC-based admission control for a multi-objective optimisation problem. **Our novelty** is in our controller design which goes **beyond the general trends** in admission control scheduling at dealing with unknown task-sets apriori and using task computation time values instead of their periods as the admission control variable.
- Justifying MPC over PID for our RTS objectives.

In conclusion to our first contribution, we affirm our thesis hypothesis for the first and second research questions. Our proposed algorithm is always better than OL admission control in managing the trade-offs between the maximisation of processor utilisation and slack values. Additionally, MPC is, indeed, better than PID with respect to managing the trade-offs between the two RTS objectives.

Our second contribution addressed the third research question in chapter 3 which is:

- Can the proposed algorithm deal with task dependencies (inside jobs) as a scheduling constraint and the emerging RTS objectives such as reducing wasted processor utilisation (WPU)? Can the proposed algorithm manage the trade-offs between the RTS objectives without prior knowledge to any RTS parameters? If so, can the proposed algorithm outperform baseline admission control algorithms such as random-based and jobs-completion-ratio-based etc.?

To the author's best knowledge, the work carried-out in chapter 5 is considered a first work in exploring an adaptive admission control algorithm for addressing task dependencies as a scheduling constraint. We have extended our initial MPC-based feedback admission control approach, proposed in chapter 4, in order to handle task dependencies and meet further RTS objectives such as the reduction of wasted processor utilisation and jobs killing ratio. We have compared our algorithm against our initial one and a number of baseline admission control algorithms. In conclusion to our second contribution, we affirm our thesis hypothesis for the third research question that our proposed feedback admission control algorithm always outperforms OL admission control baselines in managing the trade-offs between our RTS objectives.

Our third and final contribution addressed the fourth and final research question in chapter 3 which is:

- Can the algorithm support real-time rent and buy decisions for acquiring additional computing processors to support the local cluster in managing the trade-offs between the RTS objectives and increase the overall $SystemCredit$?

This question was addressed by proposing novel ERP solutions for supporting project managers in making rent and buy decisions. Such decisions are used for acquiring additional computing processors in real-time while respecting the financial constraints and incentives in its prediction and optimisation algorithms. We took advantage of control theory's ability in dealing with dynamical systems such as our industrial WMS and have extended our MPC-based feedback admission control algorithm to include two novel value-based feedback admission control algorithms. To the author's best knowledge, our work in this field is a first work towards control-theoretic ERP within the RTS context of course.

In conclusion to our third contribution, we have shown that our algorithms have shown promising results in renting and buying additional computing processors in situations where the local cluster is predicted to lack performance in meeting deadlines. We have also carried-out further experiments to show financial trade-offs between different cloud and HPC price models. Our results can give directions for project managers given certain RTS objectives and constraints that are set by them for financially justifying when to rent form the cloud or buy from HPC suppliers.

Finally, we propose our novel rent-on-demand MPC-based feedback admission control algorithm for addressing our thesis hypothesis which is:
"*Given an OL admission control baseline, can a feedback admission control algorithm improve the real-time performance in increasing the HPC cluster processor utilisation and slack values of jobs with task dependencies with abilities in extending the HPC cluster via rent and buy algorithms?*". Regardless whether an organisation uses a static or dynamic RTS policy, our proposed admission controller can regulate the workload with respect to the RTS objectives it was designed to meet via its prediction and optimisation components. Our results recommend using EDF instead of FCFS, yet our admission control algorithm can produce promising results regardless the chosen RTS policy.

7.1 Further Work

This thesis is concerned with high-level models of industrial WMSs to achieve high-level RTS objectives. One direction for further research is including input/output jitter delays and network bandwidth bottlenecks as RTS low-level constraints. However, since the work of Burkimsher in [17] has addressed such low-level RTS constraints, we propose another further work in incorporating Burkimsher's RTS policies as part of our feedback admission control work. Doing so will enable us experimenting with sophisticated RTS policies instead of traditional FCFS and EDF.

Throughout the course of this thesis the author has addressed a number of RTS issues using the proposed control-theoretic algorithm rather than focussing on one sole issue and address all the consequent issues accompanied with it. In other words, the author has addressed RTS issues in *width* rather than *depth*.

Throughout the work in chapter 4, a potential further work is to conduct system identification modelling for various workload models (low, full and high). Additionally, we can amend the FAC algorithm to include one further controller for the rejected tasks. The purpose of this extra controller is to analyse the possibilities fir re-admitting rejected tasks at a later time if the controller predicts it is feasible to so with respect to the RTS objectives.

Throughout the work in chapter 5, although we have tested our control-theoretic algorithm using typical workload models being generated from a validated workload generator depicted from Burkimsher's work in [17]. We can suggest another further work in experimenting with other industrial workload benchmarks available in the literature. Different benchmarks can include different task dependency patterns, other than the ones depicted from our industrial WMS. So, we can observe how our proposed admission controller would perform.

Throughout the work in chapter 6, we can implement dynamic assignments of job values that change over time during project's life-cycle. So, we can observe how our proposed admission controller performs with dynamic job values instead of fixed ones adopted in this thesis.

Finally, we aim to deploy our proposed feedback admission control algorithm inside our industrial WMS's HPC test-beds and observe the performance.

Appendix A

Appendix A - Periodic and independent task-sets

System	ntS_{Design}	ntS_{Data}	Processor Utilisation (%)
OL-FCFS	-13.6117	-4.3104	84.4049
-EDF	-13.1062	-2.1222	84.1808
P-FCFS	-7.7095	0.0355	68.7612
-EDF	-6.9290	0.0516	65.6101
PI-FCFS	-7.7095	0.0355	68.7612
-EDF	-6.9290	0.0516	65.6101
PID-FCFS	-7.7487	0.0281	68.5727
-EDF	-6.7647	0.0516	65.6887
MPC-FCFS	5.7896	3.6332	64.3128
-EDF	2.6723	3.4441	64.1379

Table A.1: Average values for OL, P-, PI-, PID- and MPC-based FAC algorithms with FCFS and EDF policies in a 10-processor HPC cluster with respect to Data & Design tasks *ntS* values and *PU* QoS levels. For more details on the algorithms, please refer to chapter 4.

Appendix B

Appendix B - Aperiodic and dependent task-sets

		OL		AC_t		AC_jAC_t		AC_{Random}		AC_{jSlack} & AC_{jCR}	
	Workload	**FCFS**	**EDF**	**FCFS**	**EDF**	**FCFS**	**EDF**	**FCFS**	**EDF**	**FCFS**	**EDF**
	Low	6.95	12.21	10.66	15.01	12.90	15.51	10.88	12.18	12.97	16.02
njS	*Full*	5.07	10.48	10.13	17.12	12.27	17.97	11.01	14.48	13.02	16.52
	High	3.14	8.26	12.19	14.01	13.72	14.32	13.07	12.01	13.92	13.34
$PU\%$	*Low*	76.53	92.76	76.88	91.12	79.32	93.73	77.07	94.73	74.77	91.88
	Full	77.24	94.60	76.38	93.29	77.90	94.25	74.29	95.01	78.83	92.26
	High	78.41	95.34	78.63	94.73	75.40	95.83	73.27	93.14	74.06	84.19
$WPU\%$	*Low*	NA	NA	10.06	6.08	$8e^{-3}$	0.00	10.06	4.49	88.10	42.19
	Full	NA	NA	25.03	16.02	0.19	0.00	19.12	6.49	95.16	49.19
	High	NA	NA	77.23	93.27	0.75	0.00	14.88	81.29	96.40	53.07
$jKR\%$	*Low*	NA	NA	12.00	9.01	9.00	1.38	12.00	5.15	13.00	8.02
	Full	NA	NA	30.00	34.18	8.00	4.64	34.00	17.80	47.00	32.14
	High	NA	NA	33.00	70.00	9.00	10.36	17.00	32.00	43.00	81.00

Table B.1: Results summary of admission control algorithms performance comparison in dependent soft real-time tasks. Each algorithm is tested with FCFS and EDF in a 50-processor HPC cluster. Each performance metric is experimented with low (80%), full (100%) and high (120%) workloads. For more details on the algorithms, please refer to chapter 5.

Appendix C

Appendix C - Aperiodic and dependent task-sets with ERP

		OL		AC_jAC_t		RoD £1000		RoD £500		PoD £1000		PoD £500	
	Workload	**FCFS**	**EDF**	**FCFS**	**EDF**	**FCFS**	**EDF**	**FCFS**	**EDF**	**FCFS**	**EDF**	**FCFS**	**EDF**
njS	*Low*	6.95	12.21	12.90	15.51	7.05	17.17	4.34	14.88	7.27	17.17	4.46	14.94
	Full	5.07	10.48	12.27	17.97	6.06	17.17	5.16	18.82	6.06	17.17	7.07	19.01
	High	3.14	8.26	13.72	14.32	6.08	17.17	4.05	16.21	5.16	16.69	4.34	16.56
$PU\%$	*Low*	76.53	92.76	79.32	93.73	87.12	87.50	88.32	89.63	90.34	87.49	89.59	89.91
	Full	77.24	94.60	77.90	94.25	85.83	87.28	85.66	84.53	85.83	87.28	88.42	85.32
	High	78.41	95.34	75.40	95.83	84.69	88.89	90.46	88.94	84.68	88.89	91.05	90.48
Total Savings (£)	*Low*	NA	NA	NA	NA	0.00	414.83	0.00	0.00	0.00	250	0.00	0.00
	Full	NA	NA	NA	NA	0.00	367.27	0.00	0.00	0.00	250.00	0.00	0.00
	High	NA	NA	NA	NA	0.00	456.46	0.00	0.00	0.00	250.00	0.00	0.00
$WPU\%$	*Low*	NA	NA	$8e^{-3}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Full	NA	NA	0.19	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	High	NA	NA	0.75	0.00	0.00	0.00	0.07	0.01	0.00	0.00	0.08	0.04
$jKR\%$	*Low*	NA	NA	9.00	1.38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	Full	NA	NA	8.00	4.64	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
	High	NA	NA	9.00	10.36	0.00	0.00	2.75	0.50	0.00	0.00	5.00	2.50
Total System Credit (£)	*Low*	2001.32	3031.62	2500.32	3875.58	5000.00	5414.83	5000.00	5000.00	5000.00	5250.00	5000.00	5000.00
	Full	1755.20	2893.00	2636.77	3663.60	5000.00	5367.27	5000.00	5000.00	5000.00	5250.00	5000.00	5000.00
	High	1075.27	2000.00	2060.31	2964.38	5000.00	5456.46	4685.60	4916.87	5000.00	5250.00	4173.94	4703.48

Table C.1: Results summary of admission control algorithms performance in dependent soft real-time tasks with financial incentives and constraints. Each algorithm is tested with FCFS and EDF in a 50-processor HPC cluster. Each performance metric is experimented with low (80%), full (100%) and high (120%) workloads. For more details on the algorithms, please refer to chapter 6.

Abbreviations

AC Admission Control

ANOVA ANalysis Of VaAriance

CFD Computation Fluid Dynamics

CR Completion Ratio

EDF Earliest Deadline First

ERP Enterprise Resource Planning

FAC Feedback Admission Control

FCFS First In First Out

GA Genetic Algorithms

HPC High Performance Computing

LQR Linear Quadratic Regulator

MIMO Multiple Input Multiple Output

MPC Model Predictive Control

NN Neural Networks

OL Open Loop

PID Proportional Integral Derivative

PoD Purchase on Demand

QoS Quality of Service

QP Quadratic Programming

RM Rate Monotonic

RoD Rent on Demand

RTS Real Time Systems

VAC Value-based Admission Control

WMS Workflow Management Systems

References

[1] T. F. Abdelzaher, J. A. Stankovic, C. Lu, R. Zhang, and Y. Lu, “Feedback performance control in software services,” *Control Systems, IEEE*, vol. 23, no. 3, pp. 74–90, June 2003.

[2] T. F. Abdelzaher and K. G. Shin, “Combined task and message scheduling in distributed real-time systems,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 10, no. 11, pp. 1179–1191, 1999. [Online]. Available: http://doi.ieeecomputersociety.org/10.1109/71.809575

[3] T. F. Abdelzaher, K. G. Shin, and N. Bhatti, “Performance guarantees for web server end-systems: A control-theoretical approach,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 1, pp. 80–96, Jan. 2002. [Online]. Available: http://dx.doi.org/10.1109/71.980028

[4] T. Abdelzaher and C. Lu, “Modeling and performance control of internet servers,” in *Decision and Control, 2000. Proceedings of the 39th IEEE Conference on*, vol. 3, 2000, pp. 2234–2239 vol.3.

[5] K. Ang, G. Chong, and Y. Li, “Pid control system analysis, design, and technology,” *IEEE Transactions on Control Systems Technology,*, vol. 13, no. 4, pp. 559–576, July 2005. [Online]. Available: http://eprints.gla.ac.uk/3817/

[6] J. Arnaud and S. Bouchenak, “Adaptive internet services through performance and availability control,” in *Proceedings of the 2010 ACM Symposium on Applied Computing*, ser. SAC ’10. New York, NY, USA: ACM, 2010, pp. 444–451. [Online]. Available: http://doi.acm.org/10.1145/1774088.1774182

[7] R. H. Arpaci-Dusseau and A. C. Arpaci-Dusseau, *Operating Systems: Three Easy Pieces.* Arpaci-Dusseau Books, 2014.

[8] K.-E. Årzén, A. Cervin, J. Eker, and L. Sha, “An introduction to control and scheduling co-design,” in *Proceedings of the 39th IEEE Conference on Decision and Control, 2000.*, vol. 5. IEEE, 2000, pp. 4865–4870. [Online]. Available: http://dx.doi.org/10.1109/CDC.2001.914701

[9] K.-E. Årzén, A. Robertsson, D. Henriksson, M. Johansson, H. Hjalmarsson, and K. H. Johansson, “Conclusions of the artist2 roadmap on control of computing systems,” *SIGBED Rev.*, vol. 3, no. 3, pp. 11–20, Jul. 2006. [Online]. Available: http://doi.acm.org/10.1145/1164050.1164053

[10] A. Bemporad, “Model predictive control design: New trends and tools,” in *Decision and Control, 2006 45th IEEE Conference on*, Dec 2006, pp. 6678–6683.

[11] A. Bemporad, A. Bicchi, and G. C. Buttazzo, Eds., *Hybrid Systems: Computation and Control, 10th International Workshop, HSCC 2007, Pisa, Italy, April 3-5, 2007, Proceedings*, ser. Lecture Notes in Computer Science, vol. 4416. Springer, 2007.

[12] J. Bernstein and K. McMahon, “Computing on demand - hpc as a service,” in *Penguin Computing*, 2012, pp. 1–12. [Online]. Available: http://www.penguincomputing.com/files/whitepapers/PODWhitePaper.pdf

[13] A. Bestavros and S. Nagy, “An admission control paradigm for real-time databases,” Tech. Rep., 1996.

[14] J. Broberg, S. Venugopal, and R. Buyya, “Market-oriented grids and utility computing: The state-of-the-art and future directions,” 2007.

[15] A. Burkimsher, “Dependency patterns and timing for grid workloads,” *Proceedings of the Fourth York Doctoral Symposium on Computer Science*, pp. 25–33, 2011.

[16] A. Burkimsher, I. Bate, and L. S. Indrusiak, “A survey of scheduling metrics and an improved ordering policy for list schedulers operating on workloads with dependencies and a wide variation in execution times,” *Future Generation Computer Systems*, vol. 29, no. 8, pp. 2009–2025, 2012.

[17] A. M. Burkimsher, “Fair, responsive scheduling of engineering workflows on computing grids,” Ph.D. dissertation, University of York, 2014.

[18] G. C. Buttazzo, “Rate monotonic vs. edf: Judgment day,” *Real-Time Syst.*, vol. 29, no. 1, pp. 5–26, Jan. 2005. [Online]. Available: http://dx.doi.org/10.1023/B:TIME.0000048932.30002.d9

[19] G. C. Buttazzo and L. Abeni, “Adaptive workload management through elastic scheduling.” *Real-Time Systems*, vol. 23, no. 1-2, pp. 7–24, 2002. [Online]. Available: http://dblp.uni-trier.de/db/journals/rts/rts23.html#ButtazzoA02

[20] P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini, "Efficient link scheduling for online admission control of real-time traffic in wireless mesh networks," *Computer Communications*, vol. 34, no. 8, pp. 922–934, 2011.

[21] A. Cervin, "Integrated control and real-time scheduling," Ph.D. dissertation, Lund University, 2003.

[22] K. Chen and P. Muhlethaler, "A scheduling algorithm for tasks described by time value function," *Real-Time Syst.*, vol. 10, no. 3, pp. 293–312, May 1996. [Online]. Available: http://dx.doi.org/10.1007/BF00383389

[23] R.-M. Chen and Y.-M. Huang, "Competitive neural network to solve scheduling problems," *Neurocomputing*, vol. 37, no. 1–4, pp. 177 – 196, 2001. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231200003441

[24] L. Cherkasova and P. Phaal, "Session-based admission control: a mechanism for peak load management of commercial web sites," *Computers, IEEE Transactions on*, vol. 51, no. 6, pp. 669–685, Jun 2002.

[25] D. Chillet, A. Eiche, S. Pillement, and O. Sentieys, "Real-time scheduling on heterogeneous system-on-chip architectures using an optimised artificial neural network," *Journal of Systems Architecture - Embedded Systems Design*, vol. 57, no. 4, pp. 340–353, 2011. [Online]. Available: http://dx.doi.org/10.1016/j.sysarc.2011.01.004

[26] D. Chillet, S. Pillement, and O. Sentieys, "A neural network model for real-time scheduling on heterogeneous soc architectures," in *Proceedings of the International Joint Conference on Neural Networks, IJCNN 2007, Celebrating 20 years of neural networks, Orlando, Florida, USA, August 12-17, 2007*, 2007, pp. 102–107. [Online]. Available: http://dx.doi.org/10.1109/IJCNN.2007.4370938

[27] D. C. Dhamdhere, *Operating Systems: A Concept-based Approach*, 2nd ed. Tata McGraw-Hill Education, 2006.

[28] Y. Diao, N. Gandhi, J. L. Hellerstein, S. Parekh, and D. M. Tilbury, "Using mimo feedback control to enforce policies for interrelated metrics with application to the apache web server," in *Network Operations and Management Symposium, 2002. NOMS 2002. 2002 IEEE/IFIP*, 2002, pp. 219–234. [Online]. Available: http://dx.doi.org/10.1109/NOMS.2002.1015566

[29] Y. Diao, J. L. Hellerstein, and S. Parekh, "Self-managing systems: A control theory foundation," in *In Proc of IEEE International Conference*

and Workshop on the Engineering of Computer Based Systems ECBS 2005, 2005, pp. 441–448.

[30] Y. Diao, J. L. Hellerstein, S. Parekh, R. Griffith, G. E. Kaiser, and D. Phung, “A control theory foundation for self-managing computing systems,” *IEEE journal*, vol. 23, pp. 2213–2222, 2005.

[31] R. Doraiswami, M. Stevenson, and C. Diduch, *Identification of Physical Systems: Applications to Condition Monitoring, Fault Diagnosis, Soft Sensor and Controller Design*, 1st ed. Wiley Publishing, 2014.

[32] P. Dziurzanski, H. A. Ghazzawi, and L. S. Indrusiak, “Feedback-based admission control for task allocation,” in *Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2014 9th International Symposium on*, May 2014, pp. 1–5.

[33] M. Fahmy, “A fuzzy algorithm for scheduling non-periodic jobs on soft real-time single processor system,” *Ain Shams Engineering Journal*, vol. 1, no. 1, pp. 31 – 38, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2090447910000055

[34] D. G. Feitelson, “Metrics for parallel job scheduling and their convergence,” in *Revised Papers from the 7th International Workshop on Job Scheduling Strategies for Parallel Processing*, ser. JSSPP ’01. London, UK, UK: Springer-Verlag, 2001, pp. 188–206. [Online]. Available: http://dl.acm.org/citation.cfm?id=646382.689681

[35] D. Ferrari, “Advances in real-time systems,” S. H. Son, Ed. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1995, ch. A New Admission Control Method for Real-time Communication in an Internetwork, pp. 105–116. [Online]. Available: http://dl.acm.org/citation.cfm?id=207721.207726

[36] V. Firoiu, J. Kurose, and D. Towsley, “Efficient admission control for edf schedulers,” in *INFOCOM ’97. Sixteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Driving the Information Revolution., Proceedings IEEE*, vol. 1, Apr 1997, pp. 310–317 vol.1.

[37] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 6th ed. Pearson Higher Education, Inc., 2010.

[38] A. Gambier, “Mpc and pid control based on multi-objective optimization,” in *American Control Conference, 2008*, June 2008, pp. 4727–4732.

[39] N. Gandhi, D. M. Tilbury, Y. Diao, J. Hellerstein, and S. Parekh, “MIMO control of an Apache Web server: Modeling and controller design,” in *Proceedings of the 2002 American Control Conference*, vol. 6,

2002, pp. 4922–4927. [Online]. Available: http://dx.doi.org/10.1109/ACC.2002.1025440

[40] H. A. Ghazzawi, "Scheduling approaches for large-scale complex task management," *The Proceedings of the 2nd Large Scale Complex IT Systems (LSCITS) Postgraduate Workshop*, pp. 60–66, 2010.

[41] ——, "A control-theoretic approach for scheduling soft real-time tasks with dependencies." The University of York, Tech. Rep., YCS-2014-495, 2014. [Online]. Available: https://www.cs.york.ac.uk/ftpdir/reports/2014/YCS/495/YCS-2014-495.pdf

[42] H. A. Ghazzawi, I. Bate, and L. S. Indrusiak, "A control theoretic approach for workflow management," in *Engineering of Complex Computer Systems (ICECCS), 2012 17th International Conference on.* IEEE, 2012, pp. 280–289.

[43] ——, "Mpc vs. pid controllers in multi-cpu multi-objective real-time scheduling systems," *In Proceedings of the 2012 UK Electronics Forum (UKEF'12)*, pp. 77–83, 2012.

[44] T. He, J. A. Stankovic, M. Marley, C. Lu, Y. Lu, T. F. Abdelzaher, S. Son, and G. Tao, "Feedback control-based dynamic resource management in distributed real-time systems," *J. Syst. Softw.*, vol. 80, no. 7, pp. 997–1004, Jul. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.jss.2006.09.029

[45] M. T. Helgason, "Using artificial neural networks for admission control in firm real-time systems," *Institutionen för datavetenskap*, 2000. [Online]. Available: http://www.diva-portal.org/smash/get/diva2:2784/FULLTEXT01.ps

[46] J. L. Hellerstein, Y. Diao, S. Parekh, and D. M. Tilbury, *Feedback Control of Computing Systems.* John Wiley & Sons, 2004.

[47] D. Henriksson, Y. Lu, and T. Abdelzaher, "Improved prediction for web server delay control," in *Real-Time Systems, 2004. ECRTS 2004. Proceedings. 16th Euromicro Conference on*, June 2004, pp. 61–68.

[48] D. Henriksson, A. Cervin, J. Åkesson, and K.-E. Årzén, "Feedback scheduling of model predictive controllers," in *Proceedings of the 8th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2002), 24-27 September 2002, San Jose, CA, USA*, 2002, pp. 207–216. [Online]. Available: http://dx.doi.org/10.1109/RTTAS.2002.1137395

[49] J. Heo, P. Jayachandran, I. Shin, D. Wang, T. F. Abdelzaher, and X. Liu, "Optituner: On performance composition and server

farm energy minimization application." *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 11, pp. 1871–1878, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/tpds/tpds22.html#HeoJSWAL11

[50] D. Hongjun, X. XinShun, and J. Zhiping, "A fuzzy algorithm for parallelizability evaluation and load balance on the multi-core processor," in *Granular Computing, 2008. GrC 2008. IEEE International Conference on*, Aug 2008, pp. 168–171.

[51] HP, "Cloud pricing." [Online]. Available: http://www.hpcloud.com/pricing

[52] Y.-M. Huang and R.-M. Chen, "Scheduling multiprocessor job with resource and timing constraints using neural networks." *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 29, no. 4, pp. 490–502, 1999. [Online]. Available: http://dblp.uni-trier.de/db/journals/tsmc/tsmcb29.html#HuangC99

[53] J. M. Hyman, A. A. Lazar, and G. Pacifici, "MARS: the magnet II real-time scheduling algorithm," in *SIGCOMM*, 1991, pp. 285–293. [Online]. Available: http://doi.acm.org/10.1145/115992.116018

[54] ——, "A separation principle between scheduling and admission control for broadband switching." *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 4, pp. 605–616, 1993. [Online]. Available: http://dblp.uni-trier.de/db/journals/jsac/jsac11.html#HymanLP93

[55] D. E. Irwin, L. E. Grit, and J. S. Chase, "Balancing risk and reward in a market-based task service," in *High performance Distributed Computing, 2004. Proceedings. 13th IEEE International Symposium on*, June 2004, pp. 160–169.

[56] S. Jamin, L. Zhang, D. Clark, and S. Shenker, "An admission control algorithm for predictive real-time service," in *Proceedings of the Third International Workshop on Networking and Operating System Support for Digital Audio and Video.* Springer-Verlag, New York, 1993.

[57] P. K. Janert, *Feedback Control for Computer Systems.* O'Reilly Media, Inc., 2013.

[58] E. D. Jensen, C. D. Locke, and H. Tokuda, "A time-driven scheduling model for real-time operating systems." in *RTSS*. IEEE Computer Society, 1985, pp. 112–122. [Online]. Available: http://dblp.uni-trier.de/db/conf/rtss/rtss1985.html#JensenLT85

[59] R. P. Kar and K. Porter, "Rhealstone–a real time benchmarking proposal," *Dr. Dobbs' Journal*, vol. 14, no. 2, 1989.

[60] A. S. Karuppan, B. T. Mathew, and K. Shanthakumari, "A new robust distributed real time scheduling services for rt-corba applications," in *Computing, Communication and Networking, 2008. ICCCn 2008. International Conference on*, Dec 2008, pp. 1–7.

[61] J. Kay and P. Lauder, "A fair share scheduler." *Commun. ACM*, vol. 31, no. 1, pp. 44–55, 1988. [Online]. Available: http://dblp.uni-trier.de/db/journals/cacm/cacm31.html#KayL88

[62] H. Kopetz, J. C. Laprie, B. Randell, and B. Littlewood, Eds., *Predictably Dependable Computing Systems.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 1995.

[63] K. Lai, "Markets are dead, long live markets," *SIGecom Exch.*, vol. 5, no. 4, pp. 1–10, Jul. 2005. [Online]. Available: http://doi.acm.org/10.1145/1120717.1120719

[64] S. Lauzac, R. Melhem, D. Mossé, and I. C. Society, "An improved rate-monotonic admission control and its applications," *IEEE Transactions on Computers*, vol. 52, pp. 337–350, 2003.

[65] C. Lee, J. Lehoczky, D. Siewiorek, R. Rajkumar, and J. Hansen, "A scalable solution to the multi-resource qos problem," in *Real-Time Systems Symposium, 1999. Proceedings. The 20th IEEE*, 1999, pp. 315–326.

[66] J. P. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: Exact characterization and average case behavior." in *RTSS.* IEEE Computer Society, 1989, pp. 166–171. [Online]. Available: http://dblp.uni-trier.de/db/conf/rtss/rtss1989.html#LehoczkySD89

[67] P. Li and B. Ravindran, "Fast, best-effort real-time scheduling algorithms," *Computers, IEEE Transactions on*, vol. 53, no. 9, pp. 1159–1175, Sept 2004.

[68] J. Liebeherr, D. E. Wrege, and D. Ferrari, "Exact admission control for networks with a bounded delay service," *IEEE/ACM Trans. Netw.*, vol. 4, no. 6, pp. 885–901, Dec. 1996. [Online]. Available: http://dx.doi.org/10.1109/90.556345

[69] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973. [Online]. Available: http://doi.acm.org/10.1145/321738.321743

[70] J. W. S. W. Liu, *Real-Time Systems*, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2000.

[71] L. Ljung, *System Identification: Theory for the User*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1999.

[72] C. D. Locke, "Best-effort decision making for real-time scheduling," Ph.D. dissertation, Carnegie Mellon University, 1986.

[73] C. Lu, "Feedback control real-time scheduling," Ph.D. dissertation, University of Virginia, 2001.

[74] C. Lu, Y. Lu, T. F. Abdelzaher, J. A. Stankovic, and S. H. Son, "Feedback control architecture and design methodology for service delay guarantees in web servers." *IEEE Trans. Parallel Distrib. Syst.*, vol. 17, no. 9, pp. 1014–1027, 2006. [Online]. Available: http://dblp.uni-trier.de/db/journals/tpds/tpds17.html#LuLASS06

[75] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son, "Design and evaluation of a feedback control edf scheduling algorithm," in *Real-Time Systems Symposium, 1999. Proceedings. The 20th IEEE*, 1999, pp. 56–67.

[76] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao, S. H. Son, and M. Marley, "Performance specifications and metrics for adaptive real-time systems," in *Proceedings of the 21st IEEE Conference on Real-time Systems Symposium*, ser. RTSS'10. Washington, DC, USA: IEEE Computer Society, 2000, pp. 13–23. [Online]. Available: http://dl.acm.org/citation.cfm?id=1890629.1890632

[77] C. Lu, J. A. Stankovic, S. H. Son, and G. Tao, "Feedback control real-time scheduling: Framework, modeling, and algorithms," *Journal of Real-Time Systems, Special Issue on Control-Theoretical Approaches to Real-Time Computing*, vol. 23, pp. 85–126, 2002.

[78] C. Lu, X. Wang, and X. Koutsoukos, "Feedback utilization control in distributed real-time systems with end-to-end tasks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 6, pp. 550–561, Jun. 2005. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2005.73

[79] M. Maggio, F. Terraneo, and A. Leva, "Task scheduling: A control-theoretical viewpoint for a general and flexible solution," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 4, pp. 76:1–76:22, Mar. 2014. [Online]. Available: http://doi.acm.org/10.1145/2560015

[80] L. Malrait, S. Bouchenak, and N. Marchand, "Fluid modeling and control for server system performance and availability," in *Proceedings of the 39th annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2009, [acceptance rate 20.9

[81] L. Malrait, N. Marchand, and S. Bouchenak, “Average delay guarantee in server systems using admission control,” in *Proceedings of the IFAC Workshop on Time Delay Systems (TDS)*, 2009, hal-00404404. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00404404

[82] ——, “Modeling and control of server systems: Application to database systems,” in *Proceedings of the European Control Conference (ECC)*, 2009, hal-00368537.

[83] U. ManChon, C. Ho, S. Funk, and K. Rasheed, “Gart: A genetic algorithm based real-time system scheduler,” in *Evolutionary Computation (CEC), 2011 IEEE Congress on*, June 2011, pp. 886–893.

[84] A. Masrur and S. Chakraborty, “Near-optimal constant-time admission control for dm tasks via non-uniform approximations,” in *Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Chicago, USA, 2011.

[85] Matlab, “System identification of plant models,” http://uk.mathworks.com/help/slcontrol/ug/system-identification-of-plant-models.html#bubhascl.

[86] MATLAB, *version 7.10.0 (R2010a)*. Natick, Massachusetts: The MathWorks Inc., 2010.

[87] K. R. Milliken, A. V. Cruise, R. L. Ennis, A. J. Finkel, J. L. Hellerstein, D. J. Loeb, D. A. Klein, M. J. Masullo, H. M. Van Woerkom, and W. N. B, “Yes/mvs and the autonomation of operations for large computer complexes.” in *IBM Systems Journal*, vol. 25, no. 2, 1986.

[88] M. Nicolaou, “Model predictive controllers: A critical synthesis of theory and industrial needs,” in *Advances in Chemical Engineering, Academic Press*, vol. 26, 2001, pp. 131–204.

[89] N. S. Nise, *Control Systems Engineering*, 4th ed. New York, NY, USA: John Wiley & Sons, Inc., 2003.

[90] L. Northrop, P. Feiler, R. P. Gabriel, J. Goodenough, R. Linger, T. Longstaff, R. Kazman, M. Klein, D. Schmidt, K. Sullivan, and K. Wallnau, “Ultra-Large-Scale Systems - The Software Challenge of the Future,” Software Engineering Institute, Carnegie Mellon, Tech. Rep., June 2006. [Online]. Available: http://www.sei.cmu.edu/uls/downloads.html

[91] U. of Massachusetts at Amherst, “Real-time systems: Scheduling,” http://www-ccs.cs.umass.edu/spring/sched.html.

[92] K. Ogata, *Modern Control Engineering*, 4th ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[93] Oracle, “Oracle grid engines,” http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html.

[94] ——, “Oracle data sheet: Oracle real-time scheduler for service organizations - resource planning and scheduling (rps),” Data Sheet, 2013. [Online]. Available: http://www.oracle.com/us/solutions/scm/orts-resource-planning-scheduling-1901243.pdf

[95] ——, “Oracle data sheet: Oracle real-time scheduler (ors) for service organizations„” Data Sheet, 2013. [Online]. Available: http://www.oracle.com/us/solutions/scm/ors-overview-1951196.pdf

[96] S.-M. Park and M. A. Humphrey, “Predictable high-performance computing using feedback control and admission control.” *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 3, pp. 396–411, 2011. [Online]. Available: http://dblp.uni-trier.de/db/journals/tpds/tpds22.html#ParkH11

[97] G. K. M. Pedersen and Z. Yang, “Multi-objective pid-controller tuning for a magnetic levitation system using nsga-ii,” in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO ’06. New York, NY, USA: ACM, 2006, pp. 1737–1744. [Online]. Available: http://doi.acm.org/10.1145/1143997.1144280

[98] J. M. Peha, “Scheduling and admission control for integrated-services networks: the priority token bank.” *Computer Networks*, vol. 31, no. 23-24, pp. 2559–2576, 1999. [Online]. Available: http://dblp.uni-trier.de/db/journals/cn/cn31.html#Peha99

[99] Z. Quan and J.-M. Chung, “Statistical admission control for real-time services under earliest deadline first scheduling.” *Computer Networks*, vol. 48, no. 2, pp. 137–154, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/cn/cn48.html#QuanC05

[100] A. Robertsson, B. Wittenmark, and M. Kihl, “Analysis and design of admission control in web-server systems,” in *In American Control Conference (ACC*, 2003.

[101] D. Rosu, K. Schwan, and S. Yalamanchili, “FARA - A framework for adaptive resource allocation in complex real-time systems,” in *Proceedings of the Fourth IEEE Real-Time Technology and Applications Symposium, RTAS’98, Denver, Colorado, USA, June 3-5, 1998*, 1998, pp. 79–84. [Online]. Available: http://dx.doi.org/10.1109/RTTAS.1998.683190

[102] S. R. Sakhare and D. M. S. Ali, “Genetic algorithm based adaptive scheduling algorithm for real time operating systems,” *International Journal of Embedded Systems and Applications (IJESA)*, vol. 2, no. 3, 2012.

[103] M. Saksena and S. Hong, “An engineering approach to decomposing end-to-end delays on a distributed real-time system,” in *Proc. IEEE Workshop on Parallel and Distributed Real-Time Systems*, 1996, pp. 244–251.

[104] C. Schmid and L. T. Biegler, “Quadratic programming methods for reduced hessian {SQP},” *Computers and Chemical Engineering*, vol. 18, no. 9, pp. 817 – 832, 1994, an International Journal of Computer Applications in Chemical Engineering. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0098135494E00014

[105] L. Sha, T. Abdelzaher, K.-E. Årzén, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok, “Real time scheduling theory: A historical perspective,” *Real-Time Syst.*, vol. 28, no. 2-3, pp. 101–155, Nov. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:TIME.0000045315.61234.1e

[106] D. Simon, O. Sename, D. Robert, and O. Testa, “Real-time and delay-dependent control co-design through feedback scheduling,” *CERTS Co-design in Embedded Real-time Systems*, 2003. [Online]. Available: http://necs.inrialpes.fr/people/simon/certs03.pdf

[107] J. C. Simon, *Understanding and Using Information Technology.* West Publishing Company, 1996.

[108] W. Smith, I. Foster, and V. Taylor, “Scheduling with advanced reservations,” in *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, 2000, pp. 127–132.

[109] J. A. Stankovic and K. Ramamritham, Eds., *Tutorial: Hard Real-time Systems.* Los Alamitos, CA, USA: IEEE Computer Society Press, 1989.

[110] J. A. Stankovic, K. Ramamritham, and M. Spuri, *Deadline Scheduling for Real-Time Systems: Edf and Related Algorithms.* Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[111] D. C. Steere, A. Goel, J. Gruenberg, D. Mcnamee, C. Pu, and J. Walpole, “A Feedback-driven Proportion Allocator for Real-Rate Scheduling,” in *Operating Systems Design and Implementation*, 1999, pp. 145–158. [Online]. Available: http://citeseer.ist.psu.edu/steere99feedbackdriven.html

[112] W. Sun, "A novel genetic admission control for real-time multiprocessor systems," in *The International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT '09)*, 2009, pp. 130–137.

[113] K. Suzuki, N. Uchida, H. Kuraishi, and J. Wagner, "Hpc solutions for the manufacturing industry," *FUJITSU Scientific and Technical Journal*, vol. 44, no. 4, pp. 458–566, 2008. [Online]. Available: http://www.fujitsu.com/downloads/MAG/vol44-4/paper15.pdf

[114] G. A. Tagliarini, J. F. Christ, and E. W. Page, "Optimization using neural networks," *Computers, IEEE Transactions on*, vol. 40, no. 12, pp. 1347–1358, Dec 1991.

[115] D. Trumper, "Analysis and design of feedback control systems. mit course number 2.14 / 2.140." http://web.mit.edu/2.14/www/Handouts/PoleZero.pdf, 2014.

[116] UNIVA, "Univa corporation grid engine software," http://www.univa.com/products/grid-engine, 2011.

[117] V. Vacca, F. Vasca, and L. Iannelli, "Rate admission control for hard real-time task scheduling," in *Hybrid Systems: Computation and Control, 10th International Conference*, ser. Lecture Notes on Computer Science, A. Bemporad, A. Bicchi, and G. Buttazzo, Eds. Springer-Verlag Berlin, 4 2007, vol. 4416, pp. 573–586. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71493-4

[118] O. Yang and J. Lu, "Call admission control and scheduling schemes with qos support for real-time video applications in ieee 802.16 networks." *Journal of Multimedia*, vol. 1, no. 2, pp. 21–29, 2006. [Online]. Available: http://dblp.uni-trier.de/db/journals/jmm2/jmm1.html#YangL06

[119] P. Yuan, M. Moallem, and R. Patel, "A feedback scheduling algorithm for real time control systems," in *Control Applications, 2005. CCA 2005. Proceedings of 2005 IEEE Conference on*, Aug 2005, pp. 873–878.

[120] P. Yuan, "An adaptive feedback scheduling algorithm for robot assembly and real-time control systems," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, Oct 2006, pp. 2226–2231.