A Control Theoretic Approach for Workflow Management

Hashem Ali Ghazzawi, Iain Bate, and Leandro Soares Indrusiak
The University of York, Department of Computer Science, Deramore Lane, York, YO10 5GH, UK.
{hag, iain.bate, lsi}@cs.york.ac.uk

***Abstract*—This paper explores the performance of feedback control when managing workflows in computing systems. Industrial systems nowadays can consist of geographically diverse and heterogeneous high-performance computing (HPC) clusters. When scheduling workflows over such platforms, it is often desired to observe a number of real-time objectives such as meeting deadlines, reducing slacks, and increasing platform utilisation. We apply a control theoretic approach to address scheduling-related trade-offs of workflows that are executed in HPC platforms. Our results show that model predictive control-based admission controller is efficient for scheduling periodic workflows in a homogeneous HPC cluster with respect to minimum slacks and maximum CPU utilisation.**

***Index Terms*—High-performance computing (HPC), real-time systems, scheduling, quality-of-service (QoS), feedback control, multi-input-multi-output (MIMO), admission controller (AC), system identification, model predictive control (MPC), linear quadratic regulator (LQR).**

I. Introduction

OUR research goal is to manage the trade-off in meeting the following two real-time objectives of a workflow management system that is adopted in a particular industrial organisation; (1) minimal slacks and (2) maximum CPU utilisation, where tasks are periodically generated by end-users. End-users often generate replacement workflows if previous ones do not complete within a specific time-frame. Our approach is based on carefully rejecting specific tasks that cause the scheduling performance to degrade if admitted. Problems can occur in various scenarios of industry. For instance, assume there are three tasks; tasks 1 and 2 have been generated, scheduled and are being executed in a cluster of 2 CPUs. Task 3 has been generated but still queueing to be admitted into the cluster. Task 3 has a higher probability of missing its deadline if any of tasks 1 or 2 computation time exceeds task 3's deadline value. Thus admitting task 3 irrespective to this condition can result in an increased slack for it, and hence increasing the slack of the rest of queueing tasks.

Some traditional real-time scheduling approaches assume exact knowledge of system workload and service capacity a priori. This can cause poor (1) prediction of scheduling performance in dynamical computing systems, (2) handling dynamic dependencies, and (3) managing trade-offs in multi-objective real-time optimisation [18]. This has been the motivation for alternative approaches that analyse and describe the aggregate behaviour of real-time systems.

We also see priority-driven scheduling policies such as least-laxity-first (LLF) and early-deadline-first (EDF) which are dynamic algorithms. The issue with LLF is the large overheads due to high number of context switching caused by laxity changes in real-time, hence EDF is more popular in the real-time community [18]. The issue with EDF stems from the fact that it is sometimes infeasible to guarantee sufficient compute resources due to the cost and uncertainty in today's real-time systems. Our industrial workflow management system can reach CPU overload situations and EDF's performance degrades in such scenarios [11].

One potential answer is the recent control theoretic approaches which use feedback mechanism to monitor the capacity of compute resources and quality-of-service (QoS) levels. This way, the system forces certain actions to regulate the workload for efficient scheduling performance with respect to agreed-on real-time objectives. This approach enables us handling the coupling between real-time systems components dynamically via feedback control. This is achieved by defining performance metrics e.g. CPU utilisation for transient response which can be mapped to dynamic response specifications of control systems (theory) [12]. Traditional approaches are concerned with statically assured avoidance of undesirable effects such as deadline misses and utilisation overload [13].

In this work, we are addressing a data-driven industrial workflow management system [6]. The system handles data-driven applications e.g. computational fluid dynamics (CFD) simulations, also there is no control transfer from a preceding task to the next. The workflow management system experiences poor scheduling performance with respect to the real-time objectives specified at the beginning of this section. The system consists of static scheduling based on first-come-first-served (FCFS) policy which makes missing deadlines inevitable due to the lack of dynamic (pre-emption) actions allowing higher priority workflows (with respect to deadline or resource requirements) to execute first.

Our contribution in this work is threefold. First, we implement a classical modelling approach in control theory, known as System Identification, for modelling the scheduling behaviour of the industrial workflow management system. Second, we adopt this identified model as the system (plant) model for implementing a model predictive control (MPC-)based admission controller (AC) to support real-time scheduling.

The possibility of adopting MPC to support AC is suggested in [13]. Third, we evaluate both our modelling approach and results significance towards meeting our research goal.

The paper has the following structure; Section II includes related work in the realm of control theoretic approaches in real-time scheduling. Section III includes the modelling approach we adopted in this work. Section IV shows the experimental results and their evaluations. Finally, there are conclusions and future work in Section V.

The following are the key terms used in this paper:

- **Workflow:** It has the following structure; when a task depends on (an)other task(s), they form a job (e.g. CFD simulation), and when a job depends on (an)other job(s) they form a (Data or Design) workflow. Hence, a task is the lowest level of granularity for our notion of workflows.
- **Data/Design:** They are the two types of workflows that end-users submit into the industrial workflow management system. They have different real-time scheduling attributes such as computation time and release rate.
- **Slack:** The difference between a task's deadline and its completion time.
- **Cluster CPU Utilisation:** The collective average of time each CPU spends executing a task in the cluster.
- **Simulation Framework:** It includes representative workflow generation and scheduling of jobs that mimic the ones of the industrial workflow management system.
- **Control Objectives:** Minimise tasks' slacks, and maximise cluster CPU utilisation.
- **System Inputs:** The workflow tasks computation time values of the two workflow types, Data and Design.
- **Control Variables:** They include both the monitored ***system outputs*** of the simulation framework (slacks & cluster CPU utilisation), and the ***QoS levels*** (desired slacks & cluster CPU utilisation) defined by the end-user.
- **Control Outputs:** $C_{DataMPC}$ and $C_{DesignMPC}$ are the task's ***optimal computation times*** for the two workflow types, the two values are generated by the MPC. They can also be known as the ***manipulated variables***.
- **Control Framework:** It includes both a mathematical representation of our simulation framework, Plant Model, and the proposed MPC-based AC framework.
- **Optimisation Constraints:** They include the schedulability bound and maximum/minimum values for control outputs.

II. Related Work

A classical control algorithm known as proportional-integral-derivative (PID) control has been used in uni-processor real-time systems to reduce deadline miss ratio [11]. PID is a basic mechanism to achieve control objectives in real-time systems. However when common characteristics arise such as large time delays, non-linearity, high-order dynamics, or multi-core cases, PID lacks performance improvement as it is linear and in particular symmetric [5].

MPC is an advanced method 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 [15]. In the literature, it 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 [2]. Most, if not all, computing systems exhibit non-linear behaviour [7], and the industrial workflow management system has a MIMO structure since both the number of system inputs and outputs exceeds one.

Lu chose MPC for his work in minimising the difference between the desired CPU utilisation values of *m* processors (set-points) and the actual values monitored with time [13]. The control variable was periodicity, which controls the release rate of jobs. For control tasks, there are two approaches for controlling CPU utilisation, either by manipulating the task release rates (e.g. [13]), or computation times [8], our approach is based on the latter for tasks admission. Controlling the workflow release rate is undesired in the industrial workflow management system as the distribution of submitting jobs into high-performance computing (HPC) clusters is and will not be controlled. Lu's method in controlling the periodicity of jobs can also be seen in other related work such as [16].

In the context of real-time multi-objective optimisation, there is the linear quadratic regulator (LQR) method. LQR method captures the dynamics of a system by a set of linear differential equations that can be derived via System Identification [10] e.g. the work of [4]. Diao chose LQR control algorithm in optimising CPU 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 [4]. However, it is not desired in the industrial workflow management system to force a "cap" on admitting tasks in the case of receiving one with a high priority but exceeds the maximum number of tasks specified by the controller. Our proposed AC dynamically admits/rejects tasks by following a certain admission logic using specified optimal tasks' computation time values that are computed in real-time via MPC with respect to the control objectives. This way, as long as tasks are admitted, the workflow management system behaves satisfactorily. MPC differs from LQR by allowing explicit optimisation constraints on both the system and control outputs [14].

III. Modelling

In computing systems, performance trade-offs exist such as reducing slacks while maintaining high resource utilisation [13]. Computing systems are considered non-linear, where non-linearities arise if the control objective is optimisation e.g. load balancing [7]. For instance, low deadline misses do not necessarily imply low CPU utilisation, or if we enforce a low number of jobs into a multi-processor platform, it does

not guarantee low deadline misses. In control theory these variables can be addressed as *exogenous* - meaning that they are independent of each other [10], which can be true for CPU utilisation and slacks.

Control theory relies on linear difference equations that capture the dynamics of the controlled system. A traditional modelling approach associates first-principles model of the industrial workflow management system. For example, Newton's laws are widely implemented in characterising mechanical systems. We lack such laws for computing systems, we primarily have queueing relationships etc. [7].

Using first-principles approach for our non-linear MIMO system is not only complicated, but also requires detailed knowledge of the system's dynamics especially the low-level design of the HPC servers [4]. We adopt a black-box approach that requires less details of the system dynamics as capturing them in industry can be expensive cost- and time-wise. This approach is called System Identification (or Parameter Estimation), it requires monitoring the inputs & outputs of the system we wish to control [10]. We used Matlab modelling environment, Simulink [14], for constructing our simulation framework - Figure 1 is a block representation.

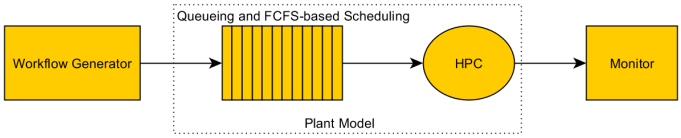

Fig. 1. Block representation of the simulation framework.

The Workflow Generator block consists of two sub-generators - one for each workflow type. There are two wait-queues - one for each sub-generater. Workflows then queue according to FCFS scheduling policy in a single run-queue to be executed in the HPC cluster.

A. Workflow Generator

We conducted ethnographic studies with end-users of the industrial workflow management system and collected the characteristics of workflow generation and scheduling behaviours. We mentioned in Section I that tasks form the lowest level of granularity of our notion of workflows. Each job consists of 10 tasks, each task is characterised by the following attributes:

- $Task_i$ is the i^{th} tasks' order, where $1 \leq i \leq 10$. There exists a dependency in the tasks-level preventing the execution of a task before its predecessor.
- C_i is the i^{th} task's computation time.
- D_i is the i^{th} task's deadline.
- T_i is the i^{th} task release rate.

The workflows task model is summarised in Table I.

Type	***C* (hours)**	***D* (hours)**	***T* (hours)**
Data	4.0-6.0 or 12.0-15.0	C+20.0%	4.0
Design	4.0-6.0	C+20.0%	0.8

TABLE I
TASK MODEL OF GENERATED JOBS.

We further elaborate our notion of deadline, D, using the terms depicted in Figure 2.

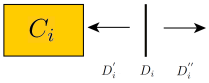

Fig. 2. Block representation of our notion of deadlines.

The deadline for all tasks (of different jobs) is set to be the computation (execution) time multiplied by an arbitrary value of 1.2 chosen for this work (that has no particular significant attribute). In other words, the deadline of individual tasks is equal to their individual computation time plus 20%. The arrows indicate the following: when the actual completion time occurs before D_i, we have a positive slack, $D_i^{'}$, otherwise negative, $D_i^{''}$.

We take the normalised value of the slacks for our results, $NormSlack_i = Slack_i/D_i$. The reason behind this is 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. If we apply the deadline-scheduling utilisation bound equation, 1, for periodic tasks, the CPU utilisation required is larger than 1 causing CPU over-utilisation.

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

Due to the static scheduling behaviour implemented in the industrial workflow management system, there is no dynamic way for scheduling tasks with respect to varying (1) slacks, and (2) CPU utilisation. Our simulation framework captures the scheduling issues arising in the industrial workflow management system. Issues include:

1) Continuous increase in deadline misses and hence in slacks due to admitting tasks without the awareness of increased queueing time in the case of CPU over-utilisation.
2) Cluster CPU under-utilisation due to the static scheduling lacking pre-emption actions with respect to deadline and/or resource utilisation priority.

Our control framework, see Figure 3, supports scheduling via its MPC-based AC approach.

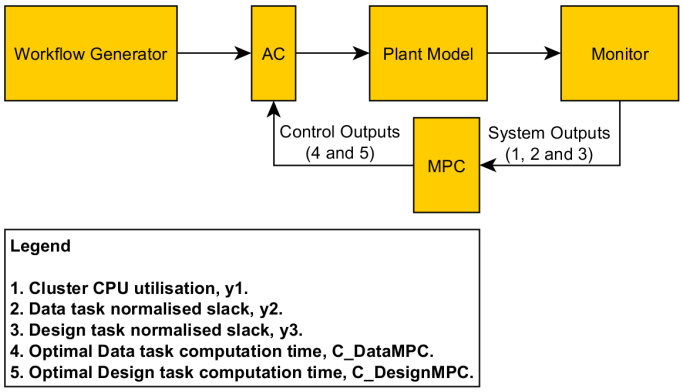

Fig. 3. Block representation of the control framework. The scheduling and execution behaviours are replaced with a Plant Model.

The MPC captures the workflow generation and scheduling characteristics via a set of linear difference equations obtained via System Identification approach. MPC maps the control objectives and QoS levels in a real-time multi-objective optimisation problem when computing the control outputs. This *problem* includes the control objectives as well as the optimisation constraints. The control outputs are then passed to the AC which uses them for its admission duty.

B. Admission Control (AC)

It is part of the proposed control framework for the industrial workflow management system. It admits tasks after receiving data from the following blocks:

1) Workflow Generator: C_{iData} & $C_{iDesign}$.
2) MPC: Optimal computation times, $C_{DataMPC}$ & $C_{DesignMPC}$. These values are computed in real-time via the MPC and are optimal with respect to the control objectives and the optimisation constraints.

The AC deals with the two workflow types in parallel, in other words it receives workflows from the sub-generators' wait-queues simultaneously. It only compares C_{iData} & $C_{iDesign}$ with $C_{DataMPC}$ & $C_{DesignMPC}$ respectively in real-time. The MPC is tuned to generate control outputs supporting the admission logic. If any task satisfies the AC logic (of its workflow type) then it is admitted into the single run-queue. If not, then that particular task is believed to degrade the scheduling performance with respect to the control objectives.

Handling rejected tasks is not present in this work. Currently in the industrial workflow management system when a particular task causes the scheduling performance to degrade, the system administrator manually *kills* the whole workflow that contains this task. This is a serious flaw in the current system. However, it is part of our research in the future to design a mechanism that allows re-admission of rejected tasks in an optimal manner with respect to the control objectives.

The logic chosen for admitting Data tasks is: ***If*** $C_{iData} \geq C_{DataMPC}$ ***Then*** *admit* $Task_{iData}$ ***Else*** *reject* $Task_{iData}$. This is due to the fact that end-users of the industrial workflow management system are satisfied with the scheduling performance of Data workflows in the current system. It was intuitive to admit as many generated Data tasks to increase the CPU utilisation. However, Data slacks have an impact on the slacks of their Design peers since they are sharing the same cluster, hence admission is restricted unless control objectives are met. The logic chosen for admitting Design tasks is: ***If*** $C_{iDesign} \leq C_{DesignMPC}$ ***Then*** *admit* $Task_{iDesign}$ ***Else*** *reject* $Task_{iDesign}$. This is due to the fact that end-users are experiencing high slack values associated with scheduling Design workflows in the current system.

C. Plant Model via System Identification

We have implemented a widely known System Identification technique called non-linear autoregressive exogenous (NARX) [10] for capturing the impact of the Workflow Generator on scheduling and execution of workflows. We implemented NARX correlation functions in our monitored system inputs and outputs. This has allowed us identifying this impact in the form of a set of transfer (system) functions in terms of complex frequency-domain representation of linear time-invariant systems. Therefore, in the control framework, Figure 3, we added these transfer functions as the Plant Model block.

An argument can be made here regarding the time-variant nature of the simulation framework. However literature of control theory supports linear time-invariant difference equations [7] [10]. We used System Identification toolbox from [14] to convert the monitored data from the simulation framework into a linear mathematical representation (preferably frequency-domain) that is compatible with control theory. More information can be found in [7]. The accuracy of our identified transfer functions have been tested against the monitored system inputs and outputs via a process called "Model Validation".

Model Validation: The Plant Model has to pass through validation test(s) before implementing it in our MPC design. Thus we evaluate how accurate does the Plant Model match the dynamics of the simulation framework, see Figure 1. In the literature and control theory we can see widely used metrics for assessing accuracy. There are (1) root-mean-square error (RMSE), (2) R^2, refer to equation 2, which computes the variability explained by the Plant Model quantifying its accuracy, and (3) correlation coefficient between the monitored data and the residuals [7].

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

In equation 2, 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, we look for models with $R^2 \geq 0.8$ [7]. The identified model has an accuracy of 82.71%.

Despite these metrics' popularity, they can be misleading. For instance, Hellerstein argues in his book that a very large R^2 can be obtained if monitored system inputs and outputs are clustered around extreme values, not necessarily reflecting on great accuracy. More information can be found in [7]. It is argued there that for validating mathematical representations of non-linear MIMO systems, residual analysis appeal more. Residuals represent the portion of validation that is not explained by the Plant Model [14]. Our residual analysis have produced plots that indicate the identified model fits within the 95% confidence interval. In other words, a good model should have a residual autocorrelation function within the confidence interval, indicating that the residuals are uncorrelated [14].

D. Model Predictive Control (MPC)

In control theoretic terms, the simulation framework is a non-linear MIMO system. Thus we require a control approach that generalises directly to such system structures. The simulation framework is also non-square, meaning it has an unequal number of control variables and outputs, and several industrial applications experience such conditions [14]. After conducting prediction and optimisation, the MPC block computes its control outputs and feeds them to the AC in order for the latter to apply its admission logic.

Prediction and Optimisation:
MPC features support for the simulation framework with its general form, see equation 3.

$$S_y(k) = \sum_{i=1}^{P} \sum_{j=1}^{n_y} \left[w_j^y [r_j(k+i) - y_j(k+i)] \right]^2 \qquad (3)$$

- P Prediction horizon = 10, which is the default value in MPC toolbox of [14].
- n_y Number of system (plant) outputs = 3.
- w_j^y Weight for system outputs y_j. We chose the values [1, 0.5, 1] for the outputs [y_1, y_2, y_3] respectively. The reason for w_1^y is to make the control framework robust with ensuring maximum possible cluster CPU utilisation. The reason for w_2^y stems from the fact, mentioned in Section III-B, that end-users of the industrial workflow management system are satisfied with the current scheduling performance of Data tasks. We reduced the MPC robustness for w_2^y i.e. Data tasks while enforced a high w_3^y i.e. Design tasks which suffer scheduling issues e.g. high positive slack values.
- r_j QoS level for each system output. We chose the values [1, 15, 15] for the outputs [y_1, y_2, y_3] respectively. This means that the desired utilisation value is 1 (i.e. 100%), and the limit of the normalised slack for both types of jobs is 15 hours. This reflects the desired values by end-users where they expect simulation jobs to complete in 1-2 working days. $r_j(k+1) - y_j(k+1)$ is the predicted deviation at future instant *k+1*.
- k Current sample interval.

The controller predicts how much each system output, y_j, deviates from its QoS level, r_j, within the prediction horizon, P. The prediction horizon refers to the number of control cycles over which the control outputs ($C_{DataMPC}$ and $C_{DesignMPC}$) are to be optimised with respect to the control objectives. MPC multiplies each deviation by each system output's weight, w_j^y, and computes the weighted sum of squared deviations, $S_y(k)$.

MPC applies equation 3 to each system output (y_1, y_2, y_3), see Figure 3, simultaneously. The sum iterations are equal to the number of system outputs, n_y. Each system output and its associated QoS level and weight are used in their particular iteration. For instance, at the second sum iteration, the MPC deals with y_2, r_2, and w_2^y. The MPC parameter values, in equation 3, are specified in Matlab environment via the MPC toolbox [14] with respect to the QoS levels, control variables and objectives.

A Quadratic Programming (QP) solver [17], which is part of MPC, deals with the optimisation problem including a quadratic objective, see equation 4.

$$\begin{aligned} J(\Delta u, \varepsilon) = \sum_{i=0}^{p-1} & \left[y(k+i+1|k) - r(k+i+1) \right]^T . \\ & Q \left[y(k+i+1|k) - r(k+i+1) \right] + \\ & \Delta u(k+i|k)^T R_{\Delta u}(k+i|k) + [u(k+i|k) - \\ & u_{\text{target}}(k+i)]^T R_u [u(k+i|k) - u_{\text{target}}(k+i)] + \rho_e \varepsilon^2 \end{aligned} \qquad (4)$$

This optimal control approach deals with the optimisation problem by minimising J, the quadratic cost function. ε is the slack variable (within MPC computational tasks). Q is an n_y by n_y matrix for the system outputs states, where n_y is the number of the system outputs. R_u is an n_u by n_u matrix for the system inputs states, where n_u is the number of the system inputs. All of these matrices are positive semi-definite and diagonal with the square of the weights, w_j^y. $u(k)$ is the signal the MPC adjusts in order to achieve its objectives, the MPC maps this value to its control outputs ($C_{DataMPC}$ and $C_{DesignMPC}$). $\Delta u(k+i|k)$ is the control signal computed at time instant $k+1$ based on the information available at time k; it follows an optimisation sequence $u(k) = u(k-1) + \Delta u(k|k-1)$.

IV. Experimental Results

In this section, we investigate performance comparisons between the proposed control framework against (simulation framework of) the industrial workflow management system with respect to the control objectives. Statistical tests were carried out signifying our results.

All of the simulations were of 200 hours period, this value was chosen to represent a week worth of jobs submission, queueing and execution of the industrial workflow management system. The set-points used in these simulations were [1, 15, 15], which are the QoS levels, r_j. The cluster sizes tested in this work include 10, 36, 50, and 100 homogenous CPUs. Please note that the control framework will be referred to as *closed-loop* (CL), and the simulation framework as *open-loop* (OL). This indicates the presence/absence of feedback control.

The following experiments were carried out; each experiment consists of three hypotheses signifying the results with respect to cluster **CPU** utilisation, **Data** slacks, and **Design** slacks respectively.

1) **First-come-first-served (FCFS) vs. early-deadline-first (EDF) scheduling**. The former is the scheduling policy adopted in the industrial workflow management system, and it is static. This experiment investigates the benefits obtained from only adopting a dynamic scheduling policy, known as EDF. The reason for testing EDF is twofold: (1) EDF is widely used in the real-time community, and (2) it is a dynamic scheduling policy that can serve the industrial workflow management system in scheduling jobs into the HPC platform.
2) **Closed-loop system (with FCFS scheduling) vs. open-loop**. Exploring the benefits for the industrial workflow management systems from only adopting the proposed control framework without changing the current scheduling policy.
3) **Closed-loop system (with EDF scheduling) vs. open-loop**. Exploring the benefits from adopting both the proposed control framework and EDF scheduling.

A. *Experiment 1: First-Come-First-Served (FCFS) vs. Early-Deadline-First (EDF) Scheduling Policies*

In the context of this experiment, we raise three hypotheses:

Hypothesis 1.1 *There is no significant difference in maximising cluster **CPU** utilisation in the industrial workflow management system when adopting EDF scheduling policy instead of FCFS.*

Table II shows the statistical summary for different test cases for cluster CPU utilisation. Note that **F** refers to FCFS scheduling and **E** for EDF. Also, the numbers next to each scheduling policy refers to the number of CPUs used for each test case.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
F10	0.00	0.82	0.89	0.92	0.94
E10	0.00	0.82	0.89	0.92	0.94
F36	0.00	0.33	0.45	0.50	0.54
E36	0.00	0.33	0.44	0.50	0.55
F50	0.00	0.27	0.33	0.37	0.40
E50	0.00	0.26	0.33	0.37	0.39
F100	0.00	0.14	0.18	0.19	0.20
E100	0.00	0.14	0.18	0.19	0.20

TABLE II
STATISTICAL SUMMARY FOR CLUSTER CPU UTILISATION OF HYPOTHESIS 1.1.

The FCFS/EDF test cases have almost the same performance with respect to cluster CPU utilisation. However, one-way analysis of variance (ANOVA) test was carried out to statistically demonstrate the difference between the two scheduling policies, Table III summarises the test.

Test Case	Mean Square	F-statistic	p-value
10 CPUs	0.01	0.42	0.52
36 CPUs	0.06	3.95	0.05
50 CPUs	0.02	2.54	0.11
100 CPUs	0.01	2.33	0.13

TABLE III
ONE-WAY ANOVA FOR CLUSTER CPU UTILISATION OF HYPOTHESIS 1.1.

The p-values for all of the test cases were ≥ 0.05. Thus, the null hypothesis is not rejected and therefore with respect to cluster CPU utilisation, there is no significant difference for the industrial workflow management system whether to adopt a dynamic scheduling policy or remain with the current static policy. This is counter-intuitive since EDF is meant to increase CPU utilisation to its maximum. However, due to the high release rate of Data and Design jobs, the cluster CPU utilisation is not affected significantly.

Hypothesis 1.2 *There is no significant difference in minimising normalised **Data** slacks in the industrial workflow management system when adopting EDF scheduling policy instead of FCFS.*

Table IV shows the statistical summary for different test cases for normalised Data slacks.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
F10	-0.33	0.24	4.58	6.69	12.54
E10	-0.33	0.49	2.00	3.27	7.06
F36	-0.44	-0.10	0.04	0.63	3.50
E36	-0.44	-0.11	0.04	0.43	1.47
F50	-0.44	-0.10	0.00	-0.13	2.25
E50	-0.44	-0.11	-0.03	0.04	0.83
F100	-0.44	-0.33	-0.10	0.00	0.21
E100	-0.44	-0.33	-0.10	0.00	0.21

TABLE IV
STATISTICAL SUMMARY FOR NORMALISED DATA SLACKS OF HYPOTHESIS 1.2.

We notice that some results overlap especially when comparing in the 100-CPUs case. This means that different scheduling policies perform differently in smaller cluster

sizes. The p-values for the first three cases (10, 36, and 50 CPUs) were found to be < 0.05 except for the 100-CPUs case where it was larger. Thus, the null hypothesis is rejected for relatively *smaller* cluster sizes. We conclude that adopting a dynamic scheduling policy in small cluster sizes (up to 50 CPUs) will significantly reduce the normalised Data slacks. However, for the industrial workflow management system, there is a number of HPC clusters that currently exceed 100 CPUs.

Hypothesis 1.3 *There is no significant difference in minimising normalised* ***Design*** *slacks in the industrial workflow management system when adopting EDF scheduling policy instead of FCFS.*

Table V shows the statistical summary for different test cases for normalised Design slacks.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
F10	0.57	8.06	14.13	20.71	34.42
E10	1.29	8.75	14.42	20.67	26.13
F36	-0.22	7.07	12.67	18.04	29.71
E36	-1.48e-16	6.38	9.71	13.88	18.04
F50	-0.22	6.50	12.63	17.74	29.71
E50	-1.48e-16	6.38	9.71	13.88	18.04
F100	-0.22	6.50	12.67	18.78	29.71
E100	-1.48e-16	6.38	9.71	13.38	18.04

TABLE V
STATISTICAL SUMMARY FOR NORMALISED DESIGN SLACKS OF HYPOTHESIS 1.3.

The p-values for the cases (36, 50, and 100 CPUs) were $<$ 0.05 and for the 10-CPUs case it was larger. Thus, the null hypothesis is rejected for relatively *larger* cluster sizes. We can conclude that adopting a dynamic scheduling policy in larger cluster sizes (above 10 CPUs) will significantly reduce the normalised Design slacks. This conclusion is fruitful in the industrial workflow management system as HPC clusters contain high number of CPUs.

Table VI summarises the benefits the industrial workflow management system can obtain from only changing the scheduling policy from being static into dynamic. Note that a '✓' in the table indicates a significant advantage in adopting EDF, and a '✗' indicates no significant difference between the two scheduling policies.

Metrics	FCFS vs. EDF
Cluster CPU Utilisation	✗.
Normalised Data Slacks (hrs)	✓small cluster sizes. ✗100 CPUs.
Normalised Design Slacks (hrs)	✓high cluster sizes. ✗10 CPUs.

TABLE VI
EXPERIMENT 1 RESULTS SUMMARY.

B. *Experiment 2: Closed-loop (with FCFS) vs. Open-loop*

In the context of this experiment, we raise three hypotheses:

Hypothesis 2.1 *There is no significant difference in maximising cluster* ***CPU*** *utilisation in the industrial workflow management system when adopting the proposed control framework.*

Table VII shows the statistical summary for different test cases for cluster CPU utilisation.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
CL10	0.00	0.54	0.76	0.80	0.83
OL10	0.00	0.82	0.89	0.92	0.94
CL36	0.00	0.29	0.45	0.50	0.54
OL36	0.00	0.33	0.45	0.50	0.54
CL50	0.00	0.26	0.34	0.37	0.40
OL50	0.00	0.27	0.33	0.37	0.40
CL100	0.00	0.14	0.18	0.19	0.20
OL100	0.00	0.14	0.18	0.19	0.20

TABLE VII
STATISTICAL SUMMARY FOR CLUSTER CPU UTILISATION OF HYPOTHESIS 2.1.

There are different performances between the two systems at smaller cluster sizes, and as we increase the number of CPUs, they almost share the same results. In fact, the table suggests that at smaller cluster sizes, the OL system enjoys higher cluster CPU utilisation than the CL system. The p-values for all the CPU cases are $= 0.05$, therefore the null hypothesis is not rejected, meaning that there is no preference between the two systems with respect to maximising cluster CPU utilisation. This is fundamentally due to the AC actions of rejecting tasks periodically in order to reduce the normalised Data and Design slacks, hence the low utilisation. For instance, let us examine the 10-CPUs case, see Figure 4.

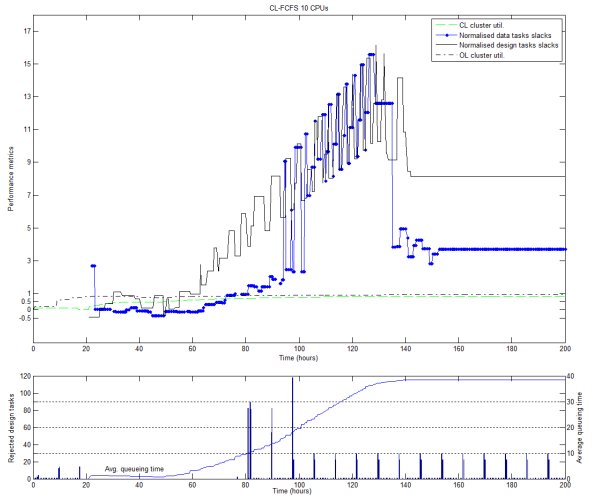

Fig. 4. Cluster CPU under-utilisation for CL-FCFS with 10 CPUs.

At the beginning of the simulation, the AC drops tasks in low number ($<$ 20 tasks) indicating the lower CPU utilisation as opposed to its OL peer where the system does not have any

AC present. Later on, the AC no longer rejects any tasks, but tasks start to queue up as their number increase due to their periodic generation. We can notice from the graph that at the 20^{th} hour, the average wait/queueing time starts to increase resulting in low cluster CPU utilisation in the CL system. Around the 80th hour we notice the following:

- Higher number of rejected tasks: 80-90.
- Higher normalised slacks.

Towards the end of the simulation, the AC maintains a rejection rate of 30 tasks per 10 hours. This explains the cluster CPU utilisation of the CL system matching its OL peer due to (1) periodic task rejections, and (2) increased queueing time. Hence, this has resulted in no significant difference between the two systems with respect to maximising cluster CPU utilisation.

Hypothesis 2.2 *There is no significant difference in minimising normalised* ***Data*** *slacks in the industrial workflow management system when adopting the proposed control framework.*

Table VIII shows the statistical summary for different test cases for normalised Data slacks.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
CL10	-0.33	0.44	3.72	4.26	15.58
OL10	-0.33	0.24	4.58	6.69	12.54
CL36	-0.44	-0.10	0.03	0.61	3.71
OL36	-0.44	-0.10	0.04	0.63	3.50
CL50	-0.44	-0.10	0.00	0.24	2.71
OL50	-0.44	-0.10	0.00	0.08	2.25
CL100	-0.44	-0.10	-0.10	0.00	2.71
OL100	-0.44	-0.33	-0.10	0.00	0.21

TABLE VIII
STATISTICAL SUMMARY FOR NORMALISED DATA SLACKS OF HYPOTHESIS 2.2.

The table above suggests that the current scheduling/queueing performance of the OL system reduces the normalised Data slacks more than the CL peer. The OL system was found to have significantly *lower* normalised Data slacks than the CL system. It is important to mention at this stage that this finding is argued from three points of view. One being the normalised Data slacks of the CL system fit in the set-point range of 15 hours, thus as far as the end-users of the industrial workflow management system are concerned, the CL system performs satisfactorily. However, it is always desired to adopt a system with lower normalised slacks. On the other hand, the MPC was tuned to be more robust towards Design jobs as they form the core of the control objectives. Finally, the CL system used in this experiment has had FCFS as its scheduling policy.

Hypothesis 2.3 *There is no significant difference in minimising normalised* ***Design*** *slacks in the industrial workflow management system when adopting the proposed control framework.*

Table IX shows the statistical summary for different test cases for normalised Design slacks.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
CL10	-0.44	2.37	8.14	8.17	16.17
OL10	0.57	8.06	14.13	20.71	34.42
CL36	-0.44	0.11	0.88	1.13	3.71
OL36	-0.22	7.07	12.67	18.04	29.71
CL50	-0.44	0.11	0.67	0.88	2.63
OL50	-0.22	6.50	12.63	17.74	29.71
CL100	-0.44	0.11	0.13	0.88	0.88
OL100	-0.22	6.50	12.67	18.78	29.71

TABLE IX
STATISTICAL SUMMARY FOR NORMALISED DESIGN SLACKS OF HYPOTHESIS 2.3.

The current scheduling/queueing performance of the OL system experiences high normalised Design slack values as opposed to its CL peer, which reduces the normalised slacks noticeably. The p-values for all the CPU cases are < 0.05, therefore the null hypothesis is rejected for all of different cluster sizes. This finding suggests the significant effect of the MPC has had on reducing the normalised Design slacks.

In summary, this experiment has shown us the benefits for the industrial workflow management system when only adopting the proposed control framework without changing the current scheduling policy. Table X summarises the findings of this experiment.

Metrics	CL-FCFS vs. OL
Cluster CPU Utilisation	✗.
Normalised Data Slacks (hrs)	✗. CL still performs within QoS limit.
Normalised Design Slacks (hrs)	✓.

TABLE X
EXPERIMENT 2 RESULTS SUMMARY.

C. *Experiment 3: Closed-loop (with EDF) vs. Open-loop*

In the context of this experiment, we raise three hypotheses:

Hypothesis 3.1 *There is no significant difference in maximising cluster* ***CPU*** *utilisation in the industrial workflow management system when adopting the proposed control framework and EDF scheduling policy.*

Table XI shows the statistical summary for different test cases for cluster CPU utilisation.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
CL10	0.00	0.55	0.76	0.79	0.83
OL10	0.00	0.82	0.89	0.92	0.94
CL36	0.00	0.28	0.45	0.50	0.54
OL36	0.00	0.33	0.45	0.50	0.54
CL50	0.00	0.25	0.33	0.37	0.40
OL50	0.00	0.29	0.33	0.37	0.40
CL100	0.00	0.14	0.18	0.19	0.20
OL100	0.00	0.14	0.18	0.19	0.20

TABLE XI
STATISTICAL SUMMARY FOR CLUSTER CPU UTILISATION OF HYPOTHESIS 3.1.

We notice different performances between the two systems at smaller cluster sizes, and as we increase the number of CPUs, they share similar results with respect to cluster CPU utilisation. In fact, the table suggests that at smaller cluster sizes, the OL system enjoys higher cluster CPU utilisation than the CL peer. The p-values for all the CPU cases are $= 0.05$, therefore the null hypothesis is not rejected. Similarly to the previous experiment, in Hypothesis 2.1, the non-significant difference stems from the increased queueing time and AC actions of rejecting tasks periodically in order to reduce the normalised Data and Design slacks. For instance, let us study the 36-CPUs case, see Figure 5:

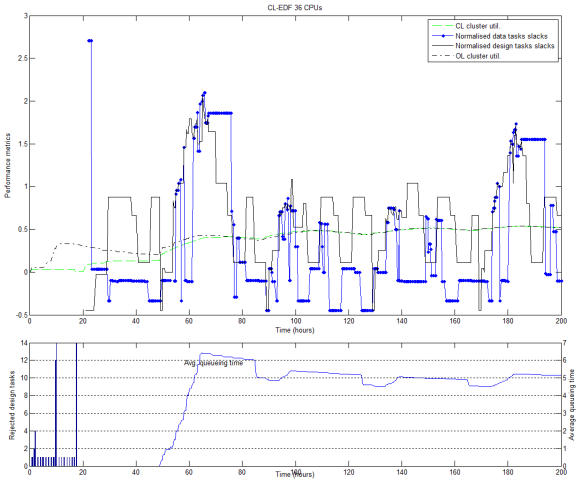

Fig. 5. Cluster CPU under-utilisation for CL-EDF with 36 CPUs.

We can see that at the beginning of the simulation, there is no queueing time present. This gives the intuition that the cluster CPU utilisation being equal or higher than its OL peer. However, due to the AC actions, the cluster CPU utilisation for the CL system is lower than its OL peer. We can notice from the graph how periodic the AC rejects tasks, which is around dropping 1 task every 1 hour, but in some instances there were higher number of rejected tasks up to 12-14 at a time. We can also notice that the AC is active only until around the 18^{th} hour of the simulation. This is because the MPC has kept the normalised slacks within the accepted range (up to 15 hours). Towards the end of the simulation, the average queueing time for tasks remain low (<6 hours), and having the AC no longer rejects any tasks, the cluster CPU utilisation of both CL and OL systems are matching. This has resulted in non-significant difference between the two systems with respect to maximising cluster CPU utilisation.

Hypothesis 3.2 *There is no significant difference in minimising normalised* ***Data*** *slacks in the industrial workflow management system when adopting the proposed control framework and EDF scheduling policy.*

Table XII shows the statistical summary for different test cases for normalised Data slacks.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
CL10	-0.33	3.10	3.54	4.33	6.86
OL10	-0.33	0.24	4.58	6.69	12.54
CL36	-0.44	-0.10	-0.01	0.72	2.71
OL36	-0.44	-0.10	0.04	0.63	3.50
CL50	-0.44	-0.11	-0.10	0.32	2.71
OL50	-0.44	-0.10	0.00	0.08	2.25
CL100	-0.44	-0.10	-0.10	0.00	2.71
OL100	-0.44	-0.33	-0.10	0.00	0.21

TABLE XII
STATISTICAL SUMMARY FOR NORMALISED DATA SLACKS OF HYPOTHESIS 3.2.

The OL system experiences lower normalised Data slacks than its CL peer. However, as we increase the number of CPUs, the CL system shares similar performance with respect to lower normalised Data slacks. The p-values for the 10/50/100-CPUs cases are < 0.05, thus the null hypothesis is rejected. The CL system significantly reduces the normalised Data slacks more than the OL system, except for the 100-CPUs case where the OL performs significantly better with respect to low normalised Data slacks. This suggests that the proposed MPC requires further fine-tuning to be robust for higher cluster sizes.

Hypothesis 3.3 *There is no significant difference in minimising normalised* ***Design*** *slacks in the industrial workflow management system when adopting the proposed control framework and EDF policy.*

Table XIII shows the statistical summary for different test cases for normalised Design slacks.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
CL10	-0.44	2.82	3.00	3.00	6.86
OL10	0.56	8.06	14.13	20.71	34.42
CL36	-0.44	0.11	0.67	0.88	2.07
OL36	-0.22	7.07	12.67	18.04	29.71
CL50	-0.44	0.11	0.67	0.88	1.37
OL50	-0.22	6.50	12.63	17.74	29.71
CL100	-0.44	0.11	0.11	0.88	0.88
OL100	-0.22	6.50	12.67	18.78	29.71

TABLE XIII
STATISTICAL SUMMARY FOR NORMALISED DESIGN SLACKS OF HYPOTHESIS 3.3.

The table suggests that the OL system experiences high normalised Design slacks as opposed to its CL peer which reduces the normalised slacks noticeably. The p-values for all of the test cases are < 0.05. Thus we reject the null hypothesis, and we can conclude that the CL system significantly reduces the normalised Design slacks more than its OL peer.

Table XIV summarises the findings of this experiment.

Metrics	CL-EDF vs. OL-FCFS
Cluster CPU Utilisation	✗.
Normalised Data Slacks (hrs)	✓10/50 CPUs.
Normalised Design Slacks (hrs)	✓.

TABLE XIV
EXPERIMENT 3 RESULTS SUMMARY.

We can conclude that adopting the proposed control framework and changing the scheduling policy of the industrial workflow management system brings a significant advantage with respect to reducing normalised Design slacks in all cases and Data slacks in two only. This is an improvement from the previous experiment with respect to normalised Data slacks, in Hypothesis 2.2, where the CL was outperformed by its OL peer for all the cases. This is a motivation for fine-tuning the control framework (the MPC in particular) to outperform the OL system in all test cases i.e. different cluster sizes.

A final remark; there is no significant difference found between FCFS & EDF scheduling policies with respect to cluster CPU utilisation, see Hypothesis 1.1. Because of that, we attempted to cover the cluster CPU under-utilisation analyses of CL system test cases with FCFS & EDF scheduling policies and the different cluster sizes across Hypotheses 2.1 and 3.1. We examined the CL-FCFS-10-CPUs case, see Figure 4, in Hypothesis 2.1 without the 36-CPUs case because the 36-CPUs case is analysed in Hypothesis 3.1, see Figure 5. The reason behind this so analysis duplicates are avoided. We did not cover analyses for the 50/100-CPUs cases because the performance observed, with respect to cluster CPU under-utilisation, matches the one of the 36-CPUs case.

V. CONCLUSIONS AND FUTURE WORK

Our work, to the authors best knowledge, is considered a contribution in the real-time control community by presenting an MPC-based AC framework for scheduling real-time systems. Our aim is to dynamically manage the trade-off between multiple real-time optimisation objectives with respect to tasks computation time values. Although this work is for a particular industrial organisation, we aim to generalise its applications to other similar workflow management systems in industry. However, our research at its current stage is in its theory phase. We look forward to implementing fair-share policies as part of our scheduling strategy in our future work allowing re-admission of rejected workflows. Also we will introduce high-level dependency trees in the jobs level introducing the notion of dependent workflows.

Our next research milestone is achieving the future research steps. We will then deploy the admission control framework into the industrial workflow management system. During this phase (of deployment) we will address classical real-time-based issues of workflow management in order to generalise our approach for the scientific communities. Such issues can include handling resource constraints like memory utilisation.

REFERENCES

[1] T. Abdelzaher, K. G. Shin and N. Bhatti. Performance Guarantees for Web Server End-Systems: A Control-Theoretical Approach. IEEE Transactions on Parallel and Distributed Systems, *13*, 80-96, 2001.

[2] T. Abdelzaher, J. Stankovic, C. Lu, R. Zhang, and Y. Lu. Feedback Performance Control in Software Services. IEEE Control Systems, *23*, 74-90, 2003.

[3] G. Buttazzo and L. Abeni. Adaptive Workload Management through Elastic Scheduling. Real-Time Systems, *23*, 7-24, 2002.

[4] Y. Diao, G. Neha, 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. The Network Operations and Management Symposiums, 219-234, 2002.

[5] G. F. Franklin, J. D. Powell, and A. Emami-Naeini. Feedback Control of Dynamic Systems, Sixth Edition. Pearson Higher Education, Inc., 2010.

[6] H. A. Ghazzawi. Scheduling Approaches for Large-Scale Complex Task Management. The Proceedings of the 2^{nd} Large Scale Complex IT Systems (LSCITS) Postgraduate Workshop, 60-66, 2010.

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

[8] D. Henriksson, A. Cervin, J. Akesson, and K.E. Arzen. Feedback Scheduling of Model Predictive Controllers. Proceedings of the Eighth IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'02), *RTAS'02*, 207-217, 2002.

[9] J. Heo, P. Jayachandran, I. Shin, D. Wang, T. Abdelzaher and X. Liu. OptiTuner: On Performance Composition and Server Farm Energy Minimization Application. IEEE Transactions on Parallel and Distributed Systems, *99*, 1045-9219, 2011.

[10] L. Ljung. System Identification: Theory for the User. Prentice-Hall, Inc., New Jersey, 1994.

[11] C. Lu, J. A. Stankovic, G. Tao and S. H. Son. Design and Evaluation of a Feedback Control EDF Scheduling Algorithm. The 20th IEEE Real-Time Systems Symposium, 56-67, 1999.

[12] C. Lu, J. A. Stankovic, T. F. Abdelzaher, G. Tao and S. Son. Performance specifications and metrics for adaptive real-time systems. Proceedings of the 21st IEEE conference on Real-time systems symposium, *RTSS'10*, 13-23, 2000.

[13] C. Lu, X. Wang and X. Koutsoukos. Feedback utilization control in distributed real-time systems with end-to-end tasks. IEEE Transactions on Parallel and Distributed Systems, *16*, 550-561, 2005.

[14] MathWorks: Matlab and Simulink for Technical Computing, 2010, http://www.mathworks.co.uk/company/?s_cid=global_nav.

[15] M. Nicolaou. Model predictive controllers: A critical synthesis of theory and industrial needs. Advances in Chemical Engineering, Academic Press, *26*, 131-204, 2001.

[16] S. M. Park and M. A. Humphrey. Predictable High-Performance Computing Using Feedback Control and Admission Control. IEEE Transactions on Parallel and Distributed Systems, *22*, 396-411, 2011.

[17] C. Schmid and L. T. Biegler. Quadratic programming methods for reduced hessian SQP. Computers & Chemical Engineering, *18*, 817-832, 1994.

[18] L. Sha, T. Abdelzaher, K. E. Arzen, 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 Systems, *28*, 101-155, 2004.