MPC vs. PID Controllers in Multi-CPU Multi-Objective Real-Time Scheduling Systems

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 provides an empirical investigation between multiple control-theoretic approaches in real-time scheduling systems. These approaches include proportional-integral-derivative (PID) and model predictive control. The former is a widely adopted controller due to its simplicity in design and implementation. The latter is considered a sophisticated one in the control community with respect to controller design. PID is widely adopted in industrial systems that are electrical and mechanical oriented. However, due to the advent of utilising control-theoretic approaches in real-time scheduling systems, sophisticated controllers have been argued to provide better performance with respect to meeting real-time multi-objective optimisation problems such as reducing deadline slacks and increasing CPU utilisation. We will also investigate multi-CPU platforms and cross view the performance of the different controllers.**

***Index Terms*—Real-time systems, scheduling, real-time multi-objective optimisation (RTMOO), quality-of-service (QoS), feedback control, multi-input-multi-output (MIMO), admission control (AC), proportional-integral-derivative (PID), model predictive control (MPC).**

I. Introduction

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 [11]. This has been the motivation for alternative approaches that analyse and describe the aggregate behaviour of real-time systems. It is not the scope of this paper to elaborate on the (dis)advantages of utilising control-theoretic approaches in real-time scheduling systems, for more details refer to [11]. We are exploring here the third point mentioned above by investigating different available control-theoretic approaches that have been adopted in the real-time community such as proportional-integral-derivative (PID) controllers e.g. [6], and model predictive control (MPC) [7].

For instance, in [6], two PID controllers were implemented to meet two real-time objectives; maximising CPU utilisation and minimising deadline miss ratio. Lu's attempt in addressing real-time multi-objective optimisation (RTMOO) has lead to significant results in his single-CPU case-study with respect to his RTMOO. In the literature we can see how PID is argued, in theory, to be used for RTMOO [2], where a mechanical industrial experimentation is illustrated in [10]. However, later on when multi-CPU scenarios were examined by Lu, MPC has offered successful results for the same RTMOO [7]. This very work of Lu has inspired us adopting MPC supporting admission control (AC) decisions for real-time scheduling, [4], as MPC-based AC was suggested in [7]. The problem statement examined in Lu's work differs from ours; the release rate of tasks (i.e. workload) cannot be controlled in our case. However, there are other related work that studies AC approaches such as [9] where controlling the workload of tasks was practiced. Shifting from PID to MPC in our work requires empirical investigation which this paper offers. The motivation for this investigation is because our problem statement involves multi-CPU platform and multi-objective optimisation problem which PID control is argued to lack performance in such scenarios [1].

Our research scope is to improve the real-time scheduling performance of a workflow management system in a particular industrial organisation [3], which is our case-study. The system handles data-driven applications e.g. CFD simulations. The system also experiences poor scheduling performance with respect to meeting real-time objectives such as meeting deadlines. 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 tasks (with respect to deadline or resource requirements) to execute first. The real-time scheduling objectives in this work are threefold. One is maximising the cluster CPU utilisation, where a cluster is considered a homogeneous high-performance computing (HPC) cluster that can be of different sizes (CPUs wise). We have two different types of tasks our case-study deals with; Data and Design. Minimising the associated deadline slacks for tasks of the different types form the second and third real-time objectives. These three objectives form the three components of our RTMOO problem.

Our contribution in this work is to offer an empirical investigation between adopting MPC- and PID-based AC approaches in the real-time-control community when dealing with periodic tasks in multi-CPU clusters with an RTMOO problem like this case-study. The structure of the paper is as follows; we introduce our case-study under the Problem Formulation section. We then discuss the Task Sets used and also both of the control frameworks (MPC- and PID-AC) for the benefit of our case-study's RTMOO. The Experimentation

section covers the results obtained signifying the performance for each control framework with respect to the RTMOO problem; there are analysis from both real-time and control theory perspectives. There will be a Summary and Discussions section, then finally we have Conclusions and Future Work.

II. Problem Formulation

The case-study we are studying in this work can be graphically represented by Figure 1.

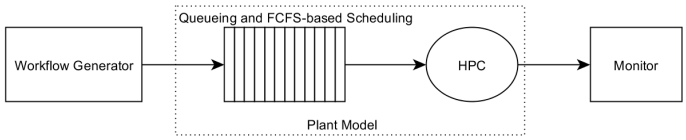

Fig. 1. Block representation of the case-study.

We can group the blocks of the above figure into three sections. One, representing the task sets used in this work. Two, representing, in control terms, the plant model of our case-study containing the queueing mechanism and tasks execution in the cluster. Three, Monitoring certain values that are used for control analysis.

A. Task Sets

Tasks form the lowest level of granularity of our notion of workflows. Each workflow consists of 10 tasks, each task is characterised by the following attributes:

Task 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.

As far as the second and third components of the RTMOO problem are concerned, we take the normalised slack values in our calculations. The reason behind that 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.

B. Plant Model

The Plant Model block refers to a linear mathematical model of queueing, scheduling and execution of the generated workflows of the industrial case-study. In brief, System Identification was used to identify the dynamic behaviour of our case-study. This assists the controller as control theory deals with linear models [5]. More details on the adopted modelling approach and its reasoning of this case-study are in [4] [1]. We proposed an MPC-based AC approach for our industrial case-study in [4], the control framework is represented in Figure 2.

[1] A copy is available at https://sites.google.com/a/york.ac.uk/hag/publications

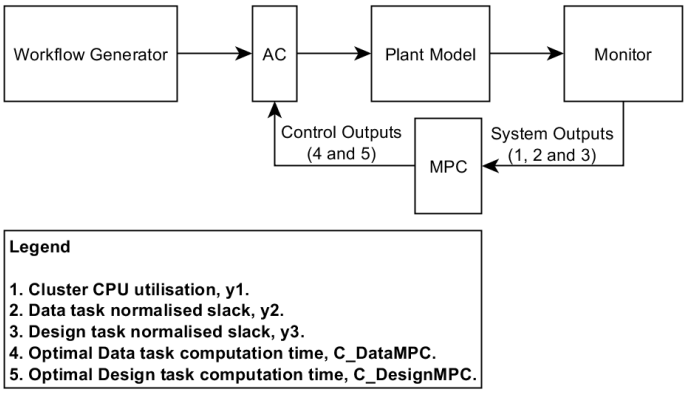

Fig. 2. Block Representation of The MPC Control Framework, [4].

When we accommodated PID control for the purpose of this paper, some modifications were done at both the AC and control blocks. For instance, we no longer have one control block (i.e. MPC), rather, there is a PID controller for each component of the RTMOO problem. This PID framework is also adopted in [6], see Figure 3.

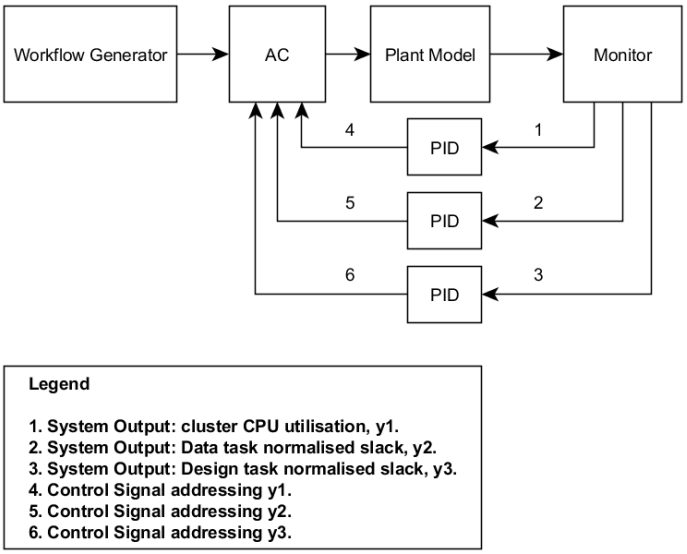

Fig. 3. Block Representation of The PID Control Framework.

The AC receives signals 4, 5, and 6 from the different PID blocks. Each control signal is computed by the PID in order to meet the desired value of each component of the RTMOO problem. For instance, signal 4 is computed to maximise cluster utilisation. At the case of receiving a Data type task, the AC maps signals 4 and 5 to an admission decision. When a Design task arrives, the AC maps signals 4 and 6 for admission control. The reason for having signal 4 consistently part of the mapping is to address the first component of our RTMOO problem. Since each control signal computed by the PID blocks are independent of each other, it was necessary to map them, accordingly, in every admission decision.

Each task has to simultaneously pass two criterions before it is fully admitted by the AC. First criteria detects if the arrived task's computation time significantly increases the system's overall slack values of that particular type (Data/Design) by multiplying the slack control signal to the system's current slack value. Second, the CPU utilisation control signal increases the likelihood of accepting a task simply in order to maximise utilisation. However in the case of CPU over-utilisation, even if a task's admission does not significantly affect the systems slack values, it will be rejected due to signal 4 mapping over-utilisation to the AC. This way, enforcing these two criterions in parallel assures that a task is accepted without the compromise of increased slack values and CPU under-/over-utilisation.

III. Experimentation

We conducted two experiments in order to cover analysis of our multiple control-theoretic approaches in the case-study. The first experiment shows real-time scheduling analysis when applying FCFS and early deadline first (EDF) policies in meeting our real-time objectives at different CPU platforms. The latter covers performance comparisons of the different controllers in the case-study. The following points summarise the settings of the experiments:

- Duration: each simulation is 500 hours representing approximately two weeks of submitting tasks into HPC clusters in the case-study.
- QoS Levels: 100% cluster CPU utilisation, and 9 hours for normalised slacks for the two types of tasks. These requirements were taken after conducting ethnographic studies in the industrial organisation.

It is important at this stage to mention that we implemented P, PI, and the full PID controllers in order to see which controller performs best under which circumstances i.e. cluster size and the different components of the RTMOO.

Significance tests are included; in the significance testing tables, a ✓refers to a significant advantage of the parameter listed next to it. For instance, "✓for FCFS in cluster CPU utilisation" means that FCFS scheduling provides significantly higher utilisation in that particular test case. Also, for instance "✓for EDF in normalised Data slacks" means that EDF provides significantly lower slack values for Data tasks in that particular test case.

A. Experiment I: Real-Time Scheduling Analysis (FCFS vs. EDF)

Each test case in the tables below refers to each system tested (**OL** for open-loop, **P**, **PI**, **PID**, and **MPC**) and the scheduling algorithm implemented (FCFS or EDF). The three tables in this section show the performance of every test case (controller and a scheduling policy) at meeting the three components of the RTMOO problem.

Table II summarises the significance testing results for cluster CPU utilisation in a 10-CPU cluster.

Test Case	Significance
OL-FCFS vs. OL-EDF	✗
P-FCFS vs. P-EDF	✓for FCFS
PI-FCFS vs. PI-EDF	✓for FCFS
PID-FCFS vs. PID-EDF	✗
MPC-FCFS vs. MPC-EDF	✗

TABLE II
Real-Time Analysis Significance Testing for Cluster CPU Utilisation in a 10-CPU Cluster.

We notice that there is no significant difference between FCFS and EDF scheduling algorithms when it comes to cluster CPU utilisation in most test-cases of this case-study. These results confirms our findings in [4]. Although this result 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 tasks, the cluster CPU utilisation is not affected significantly. It seems only P and PI controllers have experienced significantly higher utilisation with FCFS than EDF. See Figure 4 for observing the different utilisations.

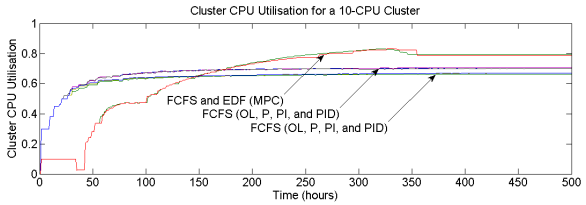

Fig. 4. Cluster CPU Utilisation of A 10-CPU Cluster.

The table below summarises the significance testing results for normalised Data slacks in a 10-CPU cluster.

Test Case	Significance
OL-FCFS vs. OL-EDF	✗
P-FCFS vs. P-EDF	✗
PI-FCFS vs. PI-EDF	✗
PID-FCFS vs. PID-EDF	✗
MPC-FCFS vs. MPC-EDF	✓for EDF

TABLE III
Real-Time Analysis Significance Testing for Normalised Data Slacks in A 10-CPU Cluster.

Across all of the controllers used, EDF scheduling proves more efficient than FCFS with MPC control with respect to reducing Data slacks. The reason there was no significant difference between FCFS and EDF across the differen controllers is because Data tasks in the industrial case-study almost always meet their deadlines on time which is depicted in our simulations. However we added Data into our RTMOO to continue respecting meeting Data tasks while working towards bringing the Design slacks significantly lower. Design slacks form the main component in our RTMOO problem.

Table IV summarises the significance testing results for normalised Design slacks in a 10-CPU cluster.

Test Case	Significance
OL-FCFS vs. OL-EDF	✓for EDF
P-FCFS vs. P-EDF	✓for EDF
PI-FCFS vs. PI-EDF	✓for EDF
PID-FCFS vs. PID-EDF	✓for EDF
MPC-FCFS vs. MPC-EDF	✓for EDF

TABLE IV
REAL-TIME ANALYSIS SIGNIFICANCE TESTING FOR NORMALISED DESIGN SLACKS IN A 10-CPU CLUSTER.

These results are intuitive as EDF is meant to respect tasks' deadline more than FCFS. Design slacks were also significantly lower with EDF in the open-loop system. The real-time analysis of the bigger cluster sizes, 50 and 100 CPUs, is covered in the Summary and Discussions section.

B. Experiment II: Control Analysis (Open-Loop vs. P vs. PI vs. PID vs. MPC)

In this experiment, there is a cross investigation of the different controllers in our case-study with respect to meeting the three components of our RTMOO. The are three tables in this section for the RTMOO components. We compare the OL system against the P controller, then the latter against PI, then the latter against PID, and finally PID against MPC. The reason behind this is because during the experimentation phase of this work, it was realised that the more sophisticated the controller, the better it performs in meeting our real-time objectives. This finding is not similar to other mechanical industries where PI control provides better performance than PID at, for instance, vibration control. For this reason, we did not include every pair comparison between the controllers.

Table V summarises the significance testing results for cluster CPU utilisation in a 10-CPU cluster.

Test Case	Significance
OL-FCFS vs. P-FCFS	✗
P-FCFS vs. PI-FCFS	✗
PI-FCFS vs. PID-FCFS	✗
PID-FCFS vs. MPC-FCFS	✓for MPC
OL-EDF vs. P-EDF	✓for P
P-EDF vs. PI-EDF	✗
PI-EDF vs. PID-EDF	✗
PID-EDF vs. MPC-EDF	✓for MPC

TABLE V
CONTROL ANALYSIS SIGNIFICANCE TESTING FOR CLUSTER CPU UTILISATION IN A 10-CPU CLUSTER.

The interesting result in the above table is that MPC outperforms PID control in maximising CPU utilisation when using any of the two scheduling algorithms. For P, PI and PID controllers, there was no significant difference amongst them with respect to utilisation. You can also refer to Figure 4 for more observations.

Table VI summarises the significance testing results for normalised Data slacks in a 10-CPU cluster.

Test Case	Significance
OL-FCFS vs. P-FCFS	✗
P-FCFS vs. PI-FCFS	✗
PI-FCFS vs. PID-FCFS	✗
PID-FCFS vs. MPC-FCFS	✓for PID
OL-EDF vs. P-EDF	✗
P-EDF vs. PI-EDF	✗
PI-EDF vs. PID-EDF	✗
PID-EDF vs. MPC-EDF	✓for PID

TABLE VI
SIGNIFICANCE TESTING FOR NORMALISED DATA SLACKS IN A 10-CPU CLUSTER.

This table shows that PID outperforms MPC in reducing Data slacks, however if we check the MPC performance, see Table XI, we notice that the results are within the QoS value of this component of the RTMOO.

The table below summarises the significance testing results for normalised Design slacks in a 10-CPU cluster.

Test Case	Significance
OL-FCFS vs. P-FCFS	✓for P
P-FCFS vs. PI-FCFS	✗
PI-FCFS vs. PID-FCFS	✗
PID-FCFS vs. MPC-FCFS	✓for MPC
OL-EDF vs. P-EDF	✓for P
P-EDF vs. PI-EDF	✗
PI-EDF vs. PID-EDF	✗
PID-EDF vs. MPC-EDF	✓for PID*

TABLE VII
SIGNIFICANCE TESTING FOR NORMALISED DESIGN SLACKS IN A 10-CPU CLUSTER.

There are two key results we can derive from the table above. One is implementing a simple P controller in the case-study proves sufficient for reducing Design slacks. The second point is when implementing FCFS, MPC proves better than PID, where the opposite is true for EDF. However, in the latter case, the MPC still performs within the QoS of this component of the RTMOO. The control analysis of the bigger cluster sizes, 50 and 100 CPUs, is covered in the Summary and Discussions section.

IV. SUMMARY AND DISCUSSIONS

As far as P, PI, and PID controllers are concerned, there was no significant incentive to adopt P or PI controls in this case-study with respect to meeting our RTMOO components as a substitute of the full PID and/or MPC. However, P and PI controllers are widely adopted in industries that are concerned with mechanical or electrical RTMOO rather than scheduling systems and resource/admission control. The reason for their popularity in other industries is for their low computation time (power) for computing control signals, please refer to [2] for more details.

Changing the control algorithm can be expensive in some industries. For industries that have the following listed real-time scheduling attributes, then considering Table VIII can be of a useful road-map if changing the scheduling algorithm is under consideration.

- Having a control-based AC framework regulating data-driven workflows into compute clusters, and/or
- Task models as in Table I.
- The three RTMOO components, and their associated QoS levels.

Test Case	OL	P or PI	PID	MPC
CPU Utilisation in 10 CPUs	FCFS	FCFS	FCFS	Either
Data Slacks in 10 CPUs	Either	Either	Either	EDF
Design Slacks in 10 CPUs	EDF	EDF	EDF	EDF
CPU Utilisation in 50 CPUs	FCFS	FCFS	FCFS	Either
Data Slacks in 50 CPUs	Either	Either	Either	EDF
Design Slacks in 50 CPUs	EDF	EDF	EDF	EDF
CPU Utilisation in 100 CPUs	FCFS	FCFS	FCFS	Either
Data Slacks in 100 CPUs	Either	Either	Either	Either
Design Slacks in 100 CPUs	EDF	EDF	EDF	Either

TABLE VIII
SUMMARY OF REAL-TIME SCHEDULING ANALYSIS.

Alternatively, if an industry is considering which controller to pick given it has the same real-time scheduling attributes listed earlier in this section, then Table IX can be a useful reference.

Test Case	FCFS	EDF
CPU Utilisation in 10 CPUs	MPC	MPC
Data Slacks in 10 CPUs	PID or MPC	PID or MPC
Design Slacks in 10 CPUs	MPC	PID or MPC
CPU Utilisation in 50 CPUs	PID	MPC
Data Slacks in 50 CPUs	PID or MPC	PID or MPC
Design Slacks in 50 CPUs	MPC	MPC
CPU Utilisation in 100 CPUs	PID	MPC
Data Slacks in 100 CPUs	MPC	MPC
Design Slacks in 100 CPUs	MPC	MPC

TABLE IX
SUMMARY OF CONTROL ANALYSIS.

As far as our case-study is concerned, and given the fact that the industrial organisation is willing to change from FCFS to EDF if proven efficient meeting all of its RTMOO components. Also, the organisation can afford adopting our proposed control-based AC approach, argued for in [4]. Then, after conducting this work, we can empirically advocate the following recommended control framework, see Table X, with respect to meeting all of the RTMOO components based on the cluster size the (organisation) utilises.

Cluster Size	Recommended Control-based AC Approach
10 CPUs	MPC with EDF
50 CPUs	MPC with EDF
100 CPUs	MPC with EDF

TABLE X
ANALYSIS SUMMARY.

At some test cases there was no significant difference between P, PI, and PID controllers with respect to some RTMOO components. However, it is a good practice to recommend a controller that has at least similar if not significantly better performance with respect to all three RTMOO components. Both EDF scheduling and MPC algorithm prove efficient with respect to all RTMOO components across the different cluster sizes.

V. CONCLUSIONS AND FUTURE WORK

We can conclude that for our industrial case-study, MPC-based AC with EDF scheduling provide, as a control framework, the most significant performance with respect to our real-time objectives. As for computer systems, this case-study is a non-linear system. Thus, changing any of the RTMOO components, cluster size, QoS levels, or the task sets can give us different results with respect to the scheduling and/or controller algorithms' performance observed in this paper. The system here is characterised with periodic tasks of different soft real-time attributes, and no disturbances were added while the simulations. Thus, we cannot recommend any of the studied approaches to case-studies with similar real-time attributes, but rather ones with the same attributes due to the non-linearity inheritance. From this statement we can derive two potential future work steps. One, is dealing with sudden change or dynamic QoS levels. Second, conduct thorough experimentations and propose a legacy control-based AC approach for similar real-time scheduling systems.

The P, PI, PID controller gain parameters i.e. each part (P/I/D) were fixed values in this work. Tuning of P/I/D parameters using software tools or classical methods such as Ziegler-Nichols method can be a potential future work. The motivation behind this is because tuning mechanisms can improve the performance of PID controllers which has been proven in many electrical and mechanical systems, [8]. It will benefit the real-time-control community in exploring PID tuning mechanisms in real-time scheduling systems. If PID proves efficient with advanced or enhanced parameter tuning mechanisms, then adopting hybrid or hierarchical control frameworks will be more plausible. This is because PID does not necessarily work on system models, and this reduces its control signal computation time significantly.

REFERENCES

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

[2] A. Gambier. MPC and PID control based on multi-objective optimization. In Proceedings of the American Control Conference, pages 47274732, 2008.

[3] 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.

[4] H. A. Ghazzawi, I. Bate, and L. S. Indrusiak. A Control-Theoretic Approach to Workflow Management. ICECCS July 2012, unpublished conference proceedings.

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

[6] 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.

[7] 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.

[8] K. Ogata. Modern Control Engineering. Prentice Hall, 2010.

[9] 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.

[10] G. K. M. Pedersen and Z. Yang. Multi-objective PID-controller tuning for a magnetic levitation system using NSGA-II. In the Proceedings of the Genetic and Evolutionary Computation Conference, 1737-1744, 2006.

[11] 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.

Appendix: Statistical Summaries.

The statistical summaries are grouped by the cluster sizes into 10, 50, and 100 CPUs i.e. three tables. Each table contains the two scheduling algorithms (FCFS and EDF) and the measured variables (cluster CPU utilisation, normalised Data slacks, and normalised Design slacks).

Note that **OL** refers to the open-loop system, **CPU** for cluster CPU utilisation, **Data** for normalised Data slacks, and **Design** for normalised Design slacks.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
OL FCFS CPU	0.0000	0.6945	0.7039	0.7069	0.7087
OL FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
OL FCFS Design	0.0000	8.5333	10.2400	12.8000	13.0000
OL EDF CPU	0.0000	0.6560	0.6640	0.6660	0.6670
OL EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
OL EDF Design	0.0000	0.2000	0.4000	0.6000	78.0800
P FCFS CPU	0.0000	0.9659	0.7053	0.7062	0.7073
P FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
P FCFS Design	0.0000	5.8000	8.4000	10.2400	13.0000
P EDF CPU	0.0000	0.6609	0.6702	0.6733	0.6749
P EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
P EDF Design	0.0000	0.2000	0.4000	0.8000	45.4400
PI FCFS CPU	0.0000	0.6959	0.7053	0.7062	0.7073
PI FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
PI FCFS Design	0.0000	5.8000	8.4000	10.2400	13.0000
PI EDF CPU	0.0000	0.6609	0.6702	0.6732	0.6748
PI EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
PI EDF Design	0.0000	0.2000	0.4000	0.8000	45.4400
PID FCFS CPU	0.0000	0.6957	0.7009	0.705	0.7071
PID FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
PID FCFS Design	0.0000	6.0000	8.3200	10.2000	13.0000
PID EDF CPU	0.0000	0.6617	0.6706	0.6735	0.6751
PID EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
PID EDF Design	0.0000	0.2000	0.4000	0.6000	44.9600
MPC FCFS CPU	0.0000	0.7954	0.7954	0.7954	0.8307
MPC FCFS Data	-0.3333	3.7222	3.7222	3.7222	15.5833
MPC FCFS Design	-0.4444	8.1389	8.1389	8.1389	16.1667
MPC EDF CPU	0.0000	0.7803	0.7867	0.7867	0.8335
MPC EDF Data	-0.3330	3.1000	3.1000	3.1000	6.8611
MPC EDF Design	-0.4440	3.0000	3.0000	3.0000	6.8611

TABLE XI
STATISTICAL SUMMARY FOR THE 10-CPU CLUSTER.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
OL FCFS CPU	0.0000	0.6945	0.7039	0.7069	0.7087
OL FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
OL FCFS Design	0.0000	8.5333	10.2400	12.8000	13.0000
OL EDF CPU	0.0000	0.6560	0.6640	0.6660	0.6670
OL EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
OL EDF Design	0.0000	0.2000	0.4000	0.6000	78.0800
P FCFS CPU	0.0000	0.1392	0.1411	0.1412	0.1415
P FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
P FCFS Design	0.0000	5.8000	8.4000	10.2400	13.0000
P EDF CPU	0.0000	0.1322	0.1340	0.1347	0.1350
P EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
P EDF Design	0.0000	0.2000	0.4000	0.8000	45.4400
PI FCFS CPU	0.0000	0.1392	0.1411	0.1412	0.1415
PI FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
PI FCFS Design	0.0000	5.8000	8.4000	10.2400	13.0000
PI EDF CPU	0.0000	0.1322	0.1340	0.1347	0.1350
PI EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
PI EDF Design	0.0000	0.2000	0.4000	0.8000	45.4400
PID FCFS CPU	0.0000	0.6958	0.7987	0.7050	0.7071
PID FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
PID FCFS Design	0.0000	6.0000	8.3200	10.2000	13.0000
PID EDF CPU	0.0000	0.1323	0.1341	0.1347	0.1350
PID EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
PID EDF Design	0.0000	0.2000	0.4000	0.6000	44.9600
MPC FCFS CPU	0.0000	0.3633	0.3940	0.4051	0.4155
MPC FCFS Data	-0.4444	-0.1026	0	-0.1944	2.7083
MPC FCFS Design	-0.4444	0.1111	0.6667	0.875	2.6250
MPC EDF CPU	0.0000	0.3566	0.3934	0.4045	0.4147
MPC EDF Data	-0.4440	-0.1070	-0.0970	0.2361	2.7083
MPC EDF Design	-0.4440	0.1111	0.5667	0.8750	1.3667

TABLE XII
STATISTICAL SUMMARY FOR THE 50-CPU CLUSTER.

Test Case	Min.	1st Q	Med.	3rd Q	Max.
OL FCFS CPU	0.0000	0.6945	0.7039	0.7069	0.7087
OL FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
OL FCFS Design	0.0000	8.5333	10.2400	12.8000	13.0000
OL EDF CPU	0.0000	0.6560	0.6640	0.6660	0.6670
OL EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
OL EDF Design	0.0000	0.2000	0.4000	0.6000	78.0800
P FCFS CPU	0.0000	0.0696	0.0705	0.0706	0.0707
P FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
P FCFS Design	0.0000	5.8000	8.4000	10.2400	13.0000
P EDF CPU	0.0000	0.0661	0.0670	0.0673	0.0674
P EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
P EDF Design	0.0000	0.2000	0.4000	0.8000	45.4400
PI FCFS CPU	0.0000	0.0696	0.0705	0.0706	0.0707
PI FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
PI FCFS Design	0.0000	5.8000	8.4000	10.24000	13.0000
PI EDF CPU	0.0000	0.0661	0.0670	0.0673	0.0675
PI EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
PI EDF Design	0.0000	0.2000	0.4000	0.8000	45.4400
PID FCFS CPU	0.0000	0.6958	0.7009	0.705	0.7071
PID FCFS Data	0.0000	0.0000	0.0000	0.0000	0.0000
PID FCFS Design	0.0000	6.0000	8.3200	10.2000	13.0000
PID EDF CPU	0.0000	0.0662	0.0671	0.0674	0.0675
PID EDF Data	0.0000	0.0000	0.0000	0.0000	0.0000
PID EDF Design	0.0000	0.2000	0.4000	0.6000	44.9600
MPC FCFS CPU	0.0000	0.1846	0.1920	0.2045	0.2093
MPC FCFS Data	-0.4440	-0.2333	-0.0972	0	2.7000
MPC FCFS Design	-0.4444	0.1111	0.1333	0.8750	0.8750
MPC EDF CPU	0.0000	0.1847	0.1991	0.2044	0.2093
MPC EDF Data	-0.4440	-0.2333	-0.0970	0	2.7030
MPC EDF Design	-0.4440	0.1111	0.2083	0.8750	0.8750

TABLE XIII
STATISTICAL SUMMARY FOR THE 100-CPU CLUSTER.