Integrating Cache Related Pre-emption Delay Analysis into EDF Scheduling

Will Lunniss[1], Sebastian Altmeyer[2], Claire Maiza[3], Robert I. Davis[1]

[1]Department of Computer Science
University of York
York, UK
{wl510,rob.davis}@york.ac.uk

[2]Department of Computer Science
Saarland University
Saarbrücken, Germany
altmeyer@cs.uni-sb.de

[3]Verimag
Grenoble INP
Grenoble, France
claire.maiza@imag.fr

Abstract — **Cache memories have been introduced into embedded systems to prevent memory access times from becoming an unacceptable performance bottleneck. Memory and cache are split into blocks containing instructions and data. During a pre-emption, blocks from the pre-empting task can evict those of the pre-empted task. When the pre-empted task is resumed, if it then has to re-load the evicited blocks, cache related pre-emption delays (CRPD) are introduced which then affect schedulability of the task. In this paper, we show how existing approaches for calculating CRPD for FP scheduling can be adapted and integrated into schedulability analysis for EDF. We then compare the performance of the different approaches against an existing approach for calculating CRPD for EDF. Using a case study and empirical evaluation, we show the benefits of our CRPD analysis.**

I. Introduction

Over the past 20 years, processor speeds have increased dramatically leaving memory access times as a major performance bottle-neck. To bridge this ever increasing gap, caches have been introduced between the processor and memory; however, they introduce significant complexity when trying to verify the timing properties of the system.

Real-time systems, especially hard real-time systems, have very stringent timing requirements and the schedulability of each task must be known. The schedulability of a taskset is determined using information about the scheduling algorithm, the arrival pattern of tasks and the tasks' *worst-case execution time*. Worst-case execution times are typically obtained assuming no pre-emption. However, in pre-emptive multi-tasking systems, caches introduce additional *cache related pre-emption delays* (CRPD) caused by the need to re-fetch blocks belonging to the pre-empted task which were evicted from cache by the pre-empting task. These CRPD effectively increase the worst-case execution time of the tasks. It is therefore important to be able to calculate, and therefore account for, CRPD when determining if a system is schedulable or not.

The motivation for this work comes from the need to provide accurate schedulability analysis for applications running on processors with cache, scheduled under EDF.

A. Related work

Earliest deadline first (EDF) is a dynamic scheduling algorithm that always schedules the job of the task with the earliest absolute deadline first. In 1973, Liu and Layland [19] gave a necessary and sufficient schedulability test for EDF for tasksets with implicit deadlines. Then in 1974, Dertouzos [12] proved EDF to be optimal among all scheduling algorithms on a uniprocessor. In 1980, Leung and Merrill [18] introduced an exact schedulability test for tasksets with constrained deadlines that was later extended to sporadic tasksets in 1990 by Baruah *et al.* [5], [6] via the *processor demand bound function.* In 2009, Zhang and Burns [24] presented their Q*uick convergence Processor-demand Analysis* (QPA) algorithm which provides an exact schedulability test for EDF which typically requires far fewer time points to be examined, than the test of Baruah *et al.* [5], [6].

Analysis of CRPD uses the concept of *useful cache blocks* (UCBs) and *evicting cache blocks* (ECBs) based on the work by Lee *et al.* [16]. ECBs are blocks that may be loaded into cache by the task during its execution. Out of the ECBs, some of them may also be UCBs. UCBs are blocks that are reused once they have been loaded into cache before potentially being evicted by the task, but not counting evictions from other pre-empting tasks. If a UCB is evicted by a pre-empting task, additional CRPD may be introduced as the UCB may have to be re-loaded when it otherwise would not have been. Depending on the approach used, the CRPD analysis combines the UCBs belonging to the pre-empted task(s) with the ECBs of the pre-empting task(s). Using this information, the total number of blocks that are evicted, which must then be reloaded after the pre-emption, can be calculated and combined with the cost of reloading a block to then give the CRPD.

A number of approaches have been developed for calculating the CRPD when using *Fixed Priority* (FP) pre-emptive scheduling. They include Lee *et al.* [16] UCB-Only approach, which considers just the pre-empted task(s), and Busquets *et al.* [11] ECB-Only approach which considers just the pre-empting task. Approaches that consider the pre-empted and pre-empting task(s) include Tan and Mooney [23] UCB-Union approach and Altmeyer *et al.* [2] ECB-Union approach. Finally, there are advanced multiset based approaches that consider the pre-empted and pre-empting task(s) by Altmeyer *et al.* [3], ECB-Union Multiset, UCB-Union Multiset, and a combined multiset approach.

There has, however, been little work towards integrating CRPD analysis into schedulability tests for dynamic scheduling algorithms. To the best of our knowledge, the only existing work on integrating CRPD analysis with EDF schedulability tests was developed by Ju *et al.* [15] in 2007.

They considered the intersection of the pre-empted task's UCBs with the pre-empting task's ECBs. However, this method for handling nested pre-emptions can lead to significant pessimism as each pair of tasks is considered separately. We discuss this method in detail at the end of Section III.

A complementary approach is to use limited pre-emption scheduling such as Bertogna and Baruah's implementation for EDF [8] from 2010. Under limited pre-emption scheduling, pre-emptions are allowed, but strongly discouraged especially when they are not required to maintain the schedulability of the system. Pre-emption can either be deferred for as long as possible, or limited to specific points in the tasks' execution. In the latter case, Bertogna *et al.* [9] presented an algorithm for optimally selecting pre-emption points out of out of a list of potential points. However, in order to limit complexity, their algorithm uses a simplified model of CRPD that only considers the pre-empted task.

In this paper, we build upon the ECB-Only, UCB-Only, UCB-Union, ECB-Union, ECB-Union Multiset, UCB-Union Multiset and combined multiset CRPD analysis for FP given by Busquets *et al.* [11], Lee *et al.* [16], Tan and Mooney [23], and Altmeyer *et al.* [2], [3], adapting them for EDF scheduling. We integrate the resulting CRPD analysis for EDF into the processor demand bound function, and hence the schedulability test for EDF given by Baruah *et al.* [5], [6].

B. Organisation

The paper is organised as follows. Section II introduces the system model, terminology and notation. Section III outlines existing EDF schedulability analysis and CRPD analysis for EDF. We then show how CRPD analysis for FP can be adapted for EDF in Section IV and V. In Section VI, we compare the performance of EDF schedulability tests incorporating the various CRPD analyses using a case study, and in Section VII using experiments based on synthetically generated tasksets. Finally, Section VIII concludes with a summary and directions for future work.

II. System Model, Terminology and Notation

This section describes the system model, terminology, and notation used in the rest of the paper.

We assume a single processor system, running a statically defined taskset under pre-emptive EDF scheduling. A taskset contains a fixed number of tasks ($\tau_1..\tau_n$) where n is a positive integer. Each task, τ_i may produce a potentially infinite stream of jobs that are separated by a minimum inter-arrival time or period T_i. Each task has a relative deadline D_i, and each job of a task has an absolute deadline d_i which is D_i after it is released. Each task has a unique task index ordered by relative deadline from smallest to largest. In the case of a tie when assigning the unique task indexes, an arbitrary choice is made. Each task also has a WCET C_i (determined for non-pre-emptive execution). In this paper, we consider tasks with *arbitrary* deadlines. (Task deadlines may be referred to as *constrained* deadlines, i.e. $D_i \leq T_i$ or *implicit* i.e. $D_i = T_i$). We assume a discrete time model. We define T_{max} as the largest period of any task in the taskset, and D_{max} as the largest relative deadline of any task in the taskset. Each task has a utilisation U_i, where $U_i = C_i / T_i$, and each taskset has a utilisation U which is equal to the sum of its tasks' utilisations.

A taskset is said to be *schedulable* with respect to a scheduling algorithm if all valid sequences of jobs generated by the taskset can be scheduled by the algorithm without any missed deadlines. A taskset is *feasible* if there exists some scheduling algorithm that can schedule all possible sequences of jobs that may be generated by the taskset without any missed deadlines. A scheduling algorithm is said to be *optimal* with respect to a task model if it can schedule all of the feasible tasksets that comply with the task model.

The EDF scheduling algorithm is an optimal dynamic scheduling algorithm for single processors which always schedules the job with the earliest absolute deadline first. Any time a job arrives with an earlier absolute deadline than the current running job, it will pre-empt the current job. When a job completes its execution, the EDF scheduler chooses the pending job with the earliest absolute deadline to execute next. In the case where two or more jobs have the same absolute deadline, we assume the scheduler always picks the job belonging to the task with the lowest unique task index, see Fig. 1. This has the benefit of minimising the number of pre-emptions. In the case where two task jobs have the same absolute and relative deadlines, it ensures that they cannot pre-empt each other. Furthermore, it ensures that after a pre-emption, the task that was pre-empted last is resumed first.

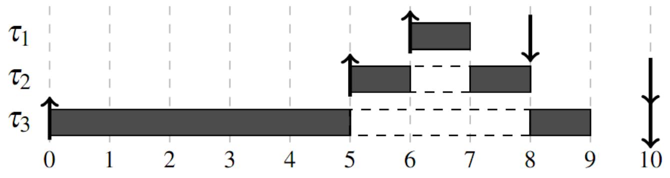

Fig. 1 – Example schedule showing how the scheduler chooses which task should execute. Task τ_3 is released at t = 0. At t = 5, task τ_2 is released, pre-empting τ_3 as although it has the same absolute deadline, it has a lower task index. At t = 6, task τ_1 is released, pre-empting task τ_2. At t = 7, τ_1 completes, the scheduler then chooses to resume task τ_2 as although it has the same absolute deadline as task τ_3, it has the lower task index.

We assume that any task τ_j with a relative deadline $D_j < D_i$ can pre-empt task τ_i. Therefore, we define the set of tasks that may have a higher priority, and can pre-empt task τ_i, as:

$$hp(i) = \{\forall \tau_j \mid D_j < D_i\} \tag{1}$$

We use $P_j(D_i)$ to denote the maximum number of times that jobs of task τ_j can pre-empt a single job of task τ_i which we calculate as follows:

$$P_j(D_i) = \max\left(0, \left\lceil \frac{D_i - D_j}{T_j} \right\rceil\right) \tag{2}$$

We use $E_j(t)$ to denote the maximum number of jobs of task τ_j that can have both their release times and their deadlines in an interval of length t, which we calculate as follows:

$$E_j(t) = \max\left(0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right) \tag{3}$$

Finally, each task τ_i has a set of UCBs, UCB$_i$ and a set of ECBs, ECB$_i$ represented by a set of integers. If for example, task τ_1 contains 4 ECBs, where the second and fourth ECBs are also UCBs, these would be represented using ECB$_1$ = {1,2,3,4} and UCB$_1$ = {2,4}.

Each time a block is reloaded, a cost is introduced that is equal to the *block reload time* (BRT).

We assume a direct mapped cache, but the work extends to set-associative caches with the LRU replacement policy as described in Section 2 of [3]. We focus on instruction only caches. In the case of data caches, the analysis would either require a write-through cache or further extension in order to be applied to write-back caches. We also assume that tasks do not share any code.

III. EDF Schedulability Analysis

In this section we recap on schedulability tests for EDF and then cover existing CRPD analysis for EDF.

A. *EDF Schedulability Tests*

In 1973, Liu and Layland [19] gave a necessary and sufficient schedulability test that indicates whether a taskset is schedulable under EDF iff $U \leq 1$, under the assumption that all tasks have implicit deadlines ($D_i = T_i$). In the case where $D_i \neq T_i$ this test is still necessary, but is no longer sufficient.

In 1974, Dertouzos [12] proved EDF to be optimal among all scheduling algorithms on a uniprocessor, in the sense that if a taskset cannot be scheduled by pre-emptive EDF, then this taskset cannot be scheduled by any algorithm.

In 1980, Leung and Merrill [18] showed that a set of periodic tasks is schedulable under EDF iff all absolute deadlines in the period [0,max{s_i}+ 2H] are met, where s_i is the start time of task τ_i, min{s_i}=0, and H is the hyperperiod (least common multiple) of all tasks' periods.

In 1990 Baruah *et al.* [5], [6] extended Leung and Merrill's work [18] to sporadic tasksets. They introduced $h(t)$, the *processor demand function*, which denotes the maximum execution time requirement of all tasks' jobs which have both their arrival times and their deadlines in a contiguous interval of length t. Using this they showed that a taskset is schedulable iff $\forall t > 0, h(t) \leq t$ where $h(t)$ is defined as:

$$h(t) = \sum_{i=1} \max\left\{0, 1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\right\} C_i \qquad (4)$$

Examining (4), it can be seen that $h(t)$ can only change when t is equal to an absolute deadline, which restricts the number of values of t that need to be checked. In order to place an upper bound on t, and therefore the number of calculations of $h(t)$, the minimum interval in which it can be guaranteed that an unschedulable taskset will be shown to be unschedulable must be found. For a general taskset with arbitrary deadlines t can be bounded by L_a [13]:

$$L_a = \max\left\{ D_1, \ldots, D_n, \frac{\sum_{i=1}^{n} (T_i - D_i) U_i}{1 - U} \right\} \qquad (5)$$

Spuri [22] and Ripoll *et al.* [21] showed that an alternative bound L_b, given by the length of the synchronous busy period can be used. Where L_b is computed by solving the following equation using fixed point iteration:

$$w^{\alpha+1} = \sum_{i=1}^{n} \left\lceil \frac{w^{\alpha}}{T_i} \right\rceil C_i \qquad (6)$$

There is no direct relationship between L_a and L_b, which enables t to be bounded by L = min(L_a, L_b). Combined with the knowledge that $h(t)$ can only change at an absolute deadline, a taskset is therefore schedulable under EDF iff $U \leq 1$ and:

$$\forall t \in Q, h(t) \leq t \qquad (7)$$

Where Q is defined as:

$$Q = \left\{ d_k \mid d_k = kT_i + D_i \wedge d_k < \min(L_a, L_b), k \in N \right\} \qquad (8)$$

In 2009, Zhang and Burns [24] presented their *Quick convergence Processor-demand Analysis* (QPA) algorithm which exploits the monotonicity of $h(t)$. QPA determines schedulability by starting with a value of t that is close to L, and then iterating back towards 0 checking a significantly smaller number of values of t than would otherwise be required.

B. *Existing CRPD Analysis*

In 2007, Ju *et al.* [15] presented an approach for integrating CRPD analysis into EDF schedulability analysis. We refer to this approach as the JCR approach after the initials of the authors' names. The JCR approach calculates the number of blocks evicted due to task τ_j directly pre-empting task τ_i multiplied by the number of times that pre-emption could occur, $P_j(D_i)$. This is repeated for each task that could pre-empt task τ_i and summed up. Using our notation, this gives the CRPD associated with task τ_i being pre-empted as follows:

$$\gamma_i^{jcr} = \text{BRT} \bullet \sum_{j \in hp(i)} P_j(D_i) \left| \text{UCB}_i \cap \text{ECB}_j \right| \qquad (9)$$

γ_i can then be integrated into (4) to give:

$$h(t) = \sum_{i=1}^{n} \max\left\{0, 1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\right\} \left(C_i + \gamma_i^{jcr} \right) \qquad (10)$$

One source of pessimism in this approach is how it deals with nested, or indirect, pre-emptions. It always defines the CRPD between a pair of tasks and adds them together. For example, if during the pre-emption of task τ_i by task τ_j, task τ_j was itself pre-empted by task τ_k the JCR approach calculates γ_i to be the sum of the pre-emptions. However, unless $\text{ECB}_j \cap \text{ECB}_k = \varnothing$ the analysis could pessimistically calculate that some UCBs are evicted multiple times.

In [15], the JCR approach was evaluated by choosing the CRPD due to task τ_j directly pre-empting task τ_i at random to be approximately 5% of C_i. When making comparisons in Section VI and VII, we use (9) for calculating the CRPD.

IV. Integrating CRPD Analysis into EDF

In this section we show how existing CRPD analysis derived for FP scheduling can be adapted for EDF scheduling and integrated into EDF schedulability tests.

In order to account for CRPD using EDF scheduling, we include a component $\gamma_{t,j}$ which represents the CRPD associated with a pre-emption by a single job of task τ_j on jobs of other tasks that are both released and have their deadlines in an interval of length t. Note, unlike its counterpart in CRPD analysis for FP scheduling, $\gamma_{t,j}$ refers to the pre-empting task τ_j and t, rather than the pre-empting and pre-empted tasks. Including $\gamma_{t,j}$ in (4) we get our revised equation for $h(t)$:

$$h(t) = \sum_{j=1}^{n} \max\left\{0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor\right\}\left(C_j + \gamma_{t,j}\right) \qquad (11)$$

In (11), we are effectively including the CRPD caused by task τ_j as if it were part of the execution time of task τ_j. Fig. 2 and Fig. 3 illustrate the CRPD increasing the execution time of the pre-empted task and modelling it as an increase in the execution time of the pre-empting task respectively.

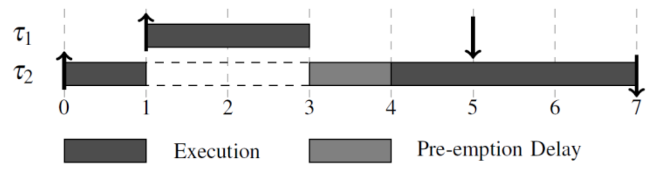

Fig. 2 – Including the CRPD caused by τ_1 pre-empting τ_2 in the execution time of τ_2

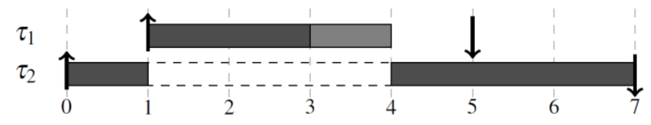

Fig. 3 – Representing the taskset in Fig. 2 by including the CRPD caused by τ_1 pre-empting τ_2 in the execution time of τ_1 which is the approach used in (11)

We make use of the approach used to prove theorem 4 in Baruah and Burns [4] to show that if a taskset is deemed schedulable by (11), e.g. Fig. 3 then the equivalent taskset which it represents, e.g. Fig. 2, is also schedulable.

Theorem 1: Let $J = \{(r_v, c_v\ d_v)\}$ denote a collection of independent jobs represented by a release time r_v execution time c_v and absolute deadline d_v. Let S be an EDF schedule of J. Let w and x be jobs of J, such that $r_w \leq r_x$ and $d_w \geq d_x$, i.e. job x is a job that pre-empts job w. Let J' be obtained from J by modifying jobs w and x to obtain jobs y and z such that $c_z = c_x - a$ and $c_y = c_w + a$ where $a \leq c_x$. (The release times and absolute deadlines of the jobs in J' are identical to their counterpart jobs in J). If J is schedulable by EDF, then so is J'.

Proof: J is equivalent to K where K is a set of sub-jobs containing c_v sub-jobs of unit length for each job v in J. Each sub-job $q_{v\,q}$ is described by ($r_{vq} = r_v$, $c_{vq} = 1$, $d_{vq} = d_v$). Let K' be a transformation of K such that a sub-jobs $q_{x\,q}$ have their deadline increased from $d_{xq} = d_x$ to d_z. Hence, K' is equivalent to J'. As S is a valid schedule for J, it is also a valid schedule for K. It follows that S is also a valid schedule for K' and hence J'. Therefore, the EDF schedule S of J proves the feasibility of J'. Since EDF is optimal on pre-emptive uniprocessors, it is therefore guaranteed to successfully schedule J' to meet all deadlines □

We need to define the set of tasks that can be pre-empted by jobs of task τ_j in an interval of length t, aff(t, j). For EDF, this set is based on the relative deadlines of the tasks. We therefore want to capture all of the tasks whose relative deadlines are greater than the relative deadline of task τ_j giving our initial definition of aff(t, j) as:

$$\mathrm{aff}(t, j) = \{\forall \tau_i \mid D_i > D_j\} \qquad (12)$$

However, we can refine this by excluding tasks whose deadlines are larger than t as they do not need to be included when calculating $h(t)$:

$$\mathrm{aff}(t, j) = \{\forall \tau_i \mid t \geq D_i > D_j\} \qquad (13)$$

as shown by Theorem 2 below.

Theorem 2: When evaluating the processor demand $h(t)$ (11) for taskset τ, the execution requirement of any task τ_k, where $D_k > t$, is not considered. Therefore, we may safely exclude any contribution to $\gamma_{t,j}$ due to the CRPD incurred by any task τ_k (where $D_k > t$) as a result of its pre-emption.

Proof: We use the proof by Baruah et al. [6] that was used to prove that (4) is necessary. Assume that taskset τ satisfies (11) and yet τ is not feasible. Let S be an EDF schedule of τ where there is a missed deadline. Let t_2 be the time of the first missed deadline and let t_1 be the last time prior to t_2 such that there is no task with a deadline $\leq t_2$ scheduled at $t_1 - 1$ in S. Since the deadline t_2 is not met, there is an active task at $t_2 - 1$, so some task must be scheduled at $t_2 - 1$. By definition of t_1 it follows that there is a task scheduled at every time in [t_1, t_2]. By the choice of t_1 and t_2, only jobs with deadlines $\leq t_2$ execute during [t_1, t_2] and all jobs released by tasks with relative deadlines $< t_2 - t_1 = t$ prior to t_1 will have completed by t_1. Therefore, as there is a task scheduled at every time in [t_1, t_2] and the deadline t_2 is missed, $h(t_2 - t_1) > t_2 - t_1$, which contradicts our original assumption that τ satisfies (11). Note in the case of a missed deadline, no job of a task τ_k with $D_k > t_2 - t_1$ executes in the interval [t_1, t_2], hence it is not necessary to include any CRPD arising in such a task □

We now show how a number of existing approaches [2] to CRPD analysis for FP scheduling can be adapted to work with EDF scheduling.

A. ECB-Only

We start with the ECB-Only approach by Busquets *et al.* [11], see Equation (3) in [3]. It captures the worst case effect of task τ_j pre-empting any task regardless of that task's UCBs, by assuming that every block evicted by task τ_j will have to be reloaded. For EDF, ECB-Only is simply:

$$\gamma_{t,j}^{ecb} = \mathrm{BRT} \bullet \left|\mathrm{ECB}_j\right| \qquad (14)$$

B. UCB-Only

The alternative UCB-Only approach by Lee *et al.* [16], see Equation (4) in [3], considers just the UCBs of the pre-empted task(s). The UCB-only approach accounts for nested pre-emptions by calculating the maximum number of UCBs that may need to be reloaded by *any* task that may be directly pre-empted by task τ_j. For EDF, this equates to the maximum number of UCBs belonging to any task that can be pre-empted by task τ_j and can also have a job with a release time and absolute deadline within an interval of length *t*. This set of tasks is given by aff(*t*, *j*), hence we can define the UCB-Only approach for EDF as:

$$\gamma_{t,j}^{ucb} = \text{BRT} \bullet \max_{\forall k \in \text{aff}(t,j)} \left\{ \left| \text{UCB}_k \right| \right\} \tag{15}$$

C. UCB-Union

The UCB-Union approach of Tan and Mooney [23], see Equation (5) in [3], accounts for the effects of nested pre-emptions by assuming that the UCBs of any tasks that could be pre-empted, including nested pre-emptions, by task τ_j are evicted by the ECBs of task τ_j. When adapting this approach for EDF, we are interested in the UCBs of any tasks that may be pre-empted by task τ_j and can also have a job with a release time and absolute deadline within an interval of length *t*. This set of tasks is again given by aff(*t*, *j*), hence, we can define the UCB-Union approach for EDF as:

$$\gamma_{t,j}^{ucb-u} = \text{BRT} \bullet \left| \left(\bigcup_{\forall k \in \text{aff}(t,j)} \text{UCB}_k \right) \cap \text{ECB}_j \right| \tag{16}$$

D. ECB-Union

The ECB-Union approach by Altmeyer *et al.* [2], see Equation (10) in [3], accounts for nested pre-emptions by making the pessimistic assumption that in any pre-emption by task τ_j, task τ_j may itself have already been pre-empted by all of the other tasks that may pre-empt it. For EDF, this set of tasks is given by $hp(j) \cup \{j\}$. Note in general this is different to the set of tasks with relative deadlines less than or equal to that of task τ_j, as tasks with the same deadline as task τ_j cannot pre-empt it. Pre-emption by task τ_j is therefore assumed to potentially evict blocks in the set $\bigcup_{h \in hp(j) \cup \{j\}} \text{ECB}_h$. The maximum number of blocks that may be evicted as a result of an already nested pre-emption by task τ_j is then obtained by considering the maximum number of UCBs that may need to be reloaded by *any* task that may be directly pre-empted by task τ_j, as in the UCB-Only case. Hence we can define the ECB-Union approach for EDF as:

$$\gamma_{t,j}^{ecb-u} = \text{BRT} \bullet \max_{\forall k \in \text{aff}(t,j)} \left\{ \left| \text{UCB}_k \cap \left(\bigcup_{h \in hp(j) \cup \{j\}} \text{ECB}_h \right) \right| \right\} \tag{17}$$

E. Effect on Task Utilisation and h(t) Calculation

We have shown how ECB-only, UCB-Only, UCB-Union, and ECB-Union CRPD analysis can be integrated into the calculation of the processor demand *h*(*t*). However, to obtain a schedulability test for EDF incorporating these CRPD analyses, we also have to adjust how we calculate task utilisation and the upper bound on the values of *t* that must be checked. Effectively, we are increasing C_j by $\gamma_{t,j}$. To account for this we introduce a modified utilisation U_j^* for task τ_j that includes the CRPD:

$$U_j^* = \frac{C_j + \gamma_{t,j}}{T_j} \tag{18}$$

We then adjust the two upper bounds for *t* by substituting U_j^* for U_j in (5) and substituting $C_j^* = C_j + \gamma_{t,j}$ for C_j in (6). (Note, when calculating $\gamma_{t,j}$ to include in C_j^* and U_j^*, we use $t = D_{\max}$, the largest relative deadline, as it gives the maximum value for $\gamma_{t,j}$).

Finally, we note that $\gamma_{t,j}$ is monotonically non-decreasing in *t* and hence using the above bounds, (11) can be used with the QPA method to obtain an efficient schedulability test for EDF scheduling accounting for CRPD. We note that this test is no longer exact as the CRPD analysis is only sufficient.

We observe that for implicit deadline tasksets, a sufficient schedulability test is simply:

$$U^* \leq 1 \tag{19}$$

V. Improved CRPD Analysis for EDF

In this section, we present improved CRPD analysis for EDF based on the multiset approaches to CRPD analysis for FP scheduling by Altmeyer *et al.* [3].

In the following analysis, we use $\gamma_{t,j}$ to represent the cost of the maximum number $E_j(t)$ of pre-emptions by jobs of task τ_j that have their release times and absolute deadlines in an interval of length *t*. It is therefore included in (4) as follows:

$$h(t) = \sum_{j=1}^{n} \left(\max\left\{ 0, 1 + \left\lfloor \frac{t - D_j}{T_j} \right\rfloor \right\} C_j + \gamma_{t,j} \right) \tag{20}$$

A. ECB-Union Multiset Approach

We now present the ECB-Union Multiset approach for EDF which is derived from the ECB-Union Multiset approach for FP scheduling, see Equations (11), (12) and (13) in Altmeyer *et al.* [3].

The ECB-Union approach is pessimistic in that it assumes that task τ_j can pre-empt any task $\tau_k \in \text{aff}(t, j)$ up to $E_j(t)$ times in an interval of length *t*. While this is potentially true if $D_k = t$, it can be a pessimistic assumption when $D_k < t$ and particularly when $D_k << T_k < t$. We can calculate a tighter bound on the number of times that jobs of task τ_k can be pre-empted by jobs of task τ_j in an interval of length *t*. This can be found by multiplying the maximum number of times task τ_j can pre-empt a single job of task τ_k, given by $P_j(D_k)$, by the number of jobs of task τ_k that are released and have their deadlines in an interval of length *t*, given by $E_k(t)$.

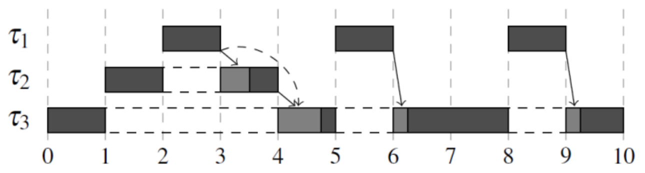

Fig. 4 – Illustration of possible pessimism with the ECB-Union approach. The pre-emption cost of task τ_1 pre-empting task τ_2 contributes three times to the total pre-emption cost of task τ_1 pre-empting other tasks in an interval of length 10; despite it only really contributing at most once.

First we form a multiset $M_{t,j}$ that contains the cost:

$$\left|\text{UCB}_k \cap \left(\bigcup_{h \in hp(j) \cup \{j\}} \text{ECB}_h\right)\right| \tag{21}$$

of task τ_j pre-empting task τ_k repeated $P_j(D_k)E_k(t)$ times, for each task $\tau_k \in \text{aff}(t, j)$, hence:

$$M_{t,j} = \bigcup_{\forall k \in \text{aff}(t,j)} \left(\bigcup_{P_j(D_k)E_k(t)} \left|\text{UCB}_k \cap \left(\bigcup_{h \in hp(j) \cup \{j\}} \text{ECB}_h\right)\right| \right) \tag{22}$$

As there are only $E_j(t)$ jobs of task τ_j with release times and deadlines in an interval of length t, the maximum CRPD is obtained by summing the $E_j(t)$ largest values in $M_{t,j}$.

$$\gamma_{t,j}^{ecb-m} = \text{BRT} \bullet \sum_{l=1}^{E_j(t)} \left| M_{t,j}^{l} \right| \tag{23}$$

Where $M_{t,j}^{l}$ is the l th largest integer value from the multiset $M_{t,j}$.

B. *UCB-Union Multiset Approach*

The UCB-Union approach is also pessimistic in that it assumes that task τ_j can pre-empt any task $\tau_k \in \text{aff}(t, j)$ up to $E_j(t)$ times. The UCB-Union Multiset approach for EDF removes this source of pessimism. It is based on the UCB-Union Multiset approach for FP scheduling, see Equations (14), (15) and (16) in Altmeyer *et al.* [3].

First we form a multiset $M_{t,j}^{ucb}$ containing $P_j(D_k)E_k(t)$ copies of the UCB$_k$ of each task $\tau_k \in \text{aff}(t, j)$. This multiset reflects the fact that jobs of task τ_j cannot evict the UCBs of jobs of task τ_k that have both their release times and deadlines in an interval of length t more than $P_j(D_k)E_k(t)$ times. Hence:

$$M_{t,j}^{ucb} = \bigcup_{\forall k \in \text{aff}(t,j)} \left(\bigcup_{P_j(D_k)E_k(t)} \text{UCB}_k \right) \tag{24}$$

Next we form a multiset $M_{t,j}^{ecb}$ containing $E_j(t)$ copies of the ECB$_j$ of task τ_j. This multiset reflects the fact that there are at most $E_j(t)$ jobs of task τ_j that have their release times and deadlines in an interval of length t, each of which can evict ECBs in the set ECB$_j$.

$$M_{t,j}^{ecb} = \bigcup_{E_j(t)} \left(\text{ECB}_j\right) \tag{25}$$

$\gamma_{t,j}^{ucb-m}$ is then given by the size of the multi-set intersection between $M_{t,j}^{ucb}$ and $M_{t,j}^{ecb}$:

$$\gamma_{t,j}^{ucb-m} = \text{BRT} \bullet \left| M_{t,j}^{ucb} \cap M_{t,j}^{ecb} \right| \tag{26}$$

C. *Combined Multiset Approach*

The ECB-Union Multiset and UCB-Union Multiset approaches are incomparable, we can therefore calculate $h(t)$ at each stage of the QPA algorithm using both approaches and take the minimum to form a combined approach:

$$h(t) = \min\left(h(t)^{ucb-m}, h(t)^{ecb-m} \right) \tag{27}$$

D. *Effect on Task Utilisation and h(t) Calculation*

The multiset approaches calculate the CRPD for all of the tasks in one go. Therefore, inflating the upper bounds on t used in the schedulability test (5), (6) by substituting in U^* and C^* as described in Section IV.E is not possible. This is because the test that $U^* \leq 1$ may pass even though one or more tasks may have utilisations > 1, causing them to miss a deadline. Therefore, we need a new upper bound.

The method we use to determine a suitable upper bound is based on using an upper bound on the utilisation due to CRPD that is valid for all intervals of length greater than some value L_c. We then use this CRPD utilisation value to inflate the taskset utilisation and thus compute an upper bound L_d on the maximum length of the synchronous busy period. This upper bound is valid provided that it is greater than L_c, otherwise the actual maximum length of the busy period may lie somewhere in the interval $[L_d, L_c]$, hence we can use $\max(L_c, L_d)$ as a bound.

We choose a value of $t = L_c = 100\ T_{max}$ which limits the overestimation of the CRPD utilisation $U^{\gamma} = \gamma_t / t$ to at most 1%. We then calculate γ_t using (23) for ECB-Union Multiset and (26) for UCB-Union Multiset. However, in (22) and (24) & (25), we substitute $E_x^{max}(t)$ for $E_x(t)$ to ensure that the computed value of U^{γ} is a valid upper bound for all intervals of length $t \geq L_c$.

$$E_x^{max}(t) = \max\left(0, 1 + \left\lceil \frac{t - D_x}{T_x} \right\rceil \right) \tag{28}$$

We then check that $U + U^{\gamma} < 1$, if not then we deem the taskset unschedulable, otherwise we compute an upper bound on the length of the busy period via a modified version of (3):

$$w^{\alpha+1} \leq \sum_{\forall j} \left(\frac{w^{\alpha}}{T_j} + 1 \right) C_j + w^{\alpha} U^{\gamma} \tag{29}$$

rearranged to give:

$$w \leq \frac{1}{\left(1 - \left(U + U^{\gamma}\right)\right)} \sum_{\forall j} U_j T_j \tag{30}$$

Then, substituting in T_{max} for each value of T_j we get our upper bound:

$$L_d = \frac{U \bullet T_{max}}{\left(1 - \left(U + U^{\gamma}\right)\right)} \tag{31}$$

We then use $L = \max(L_c, L_d)$ as the maximum value of t to check in the EDF schedulability test.

E. *Comparability and Dominance*

The CRPD analyses for EDF scheduling have similar comparability relationships to their counterparts presented in [3] for FP scheduling. The UCB-Union approach dominates the ECB-Only approach, and the ECB-Union approach dominates the UCB-Only approach. The JCR approach is incomparable with all of the non-multiset approaches. However, if we re-write the JCR approach (9) so that it calculates the cost of all $E_j(t)$ pre-emptions at once, then it can be seen that the UCB-Union Multiset approach dominates it.

$$\gamma_{t,i}^{jcr} = \mathrm{BRT} \bullet \sum_{\forall j \in D_i \geq D_j \wedge i \neq j} P_j(D_i) E_j(t) \left| \mathrm{UCB}_i \cap \mathrm{ECB}_j \right| \quad (32)$$

Furthermore, the UCB-Union Multiset approach dominates the UCB-Union approach and the ECB-Union Multiset approach dominates the ECB-Union approach. This is because the sum of the $E_j(t)$ largest pre-emption costs will always be less than or equal to $E_j(t)$ multiplied by the largest pre-emption cost. The combined multiset approach dominates all other approaches as shown in Fig. 5. Furthermore, because the combined approach uses the two multiset approaches at each stage of the QPA algorithm, the number of tasksets that it deems schedulable can is greater than a simple union of the two multiset approaches.

We note that including the CRPD as if it were additional execution time of the pre-empting task, as we have done in all of the non-multiset approaches, has the potential for significant pessimism if the execution time of a task τ_i is close to its deadline such that:

$$C_j \leq D_j < C_j + \gamma_{t,j} \quad (33)$$

In this case task τ_i would be deemed unschedulable when it may not be. This problem is avoided by the multiset approaches.

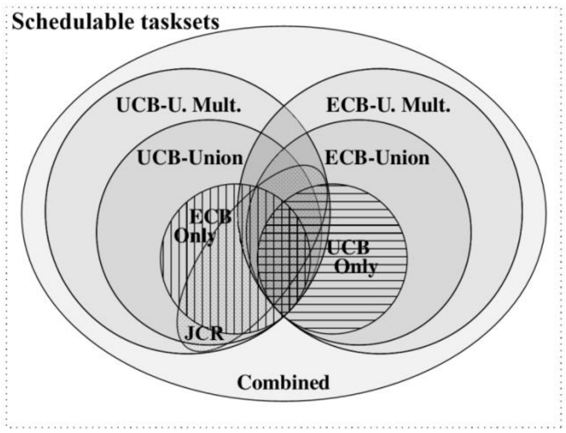

Fig. 5 - Venn diagram illustrating the relationship between the different approaches used to calculate CRPD. The larger the area, the more tasksets deemed schedulable by the approach.

I. Case Study

In this section we evaluate the schedulability tests for EDF including integrated CRPD analysis using the approaches introduced in this paper: ECB-Only, UCB-Only, UCB-Union, ECB-Union, ECB-Union Multiset, UCB-Union Multiset and the combined multiset approaches, as well as the JCR approach of Ju *et al.* [15] on a case study. For comparison purposes, we also used the EDF schedulability test assuming no pre-emption costs.

The case study is the same one used in Altmeyer *et al* [2] to evaluate CRPD analysis for systems using FP scheduling. The case study comprises a number of tasks from the Mälardalen benchmark suite[1] [14]. While these tasks do not represent a real taskset, they do represent typical code found in real-time systems. For each task, the WCET and number of ECBs and UCBs are taken from [1], details for each task can be found in Table 1 of [2], and [3]. The system was setup to model an ARM7 processor[2] clocked at 10MHz with a 2KB direct-mapped instruction cache with a line size of 8 Bytes giving 256 cache sets, 4 Byte instructions, and a BRT of 8μs.

The taskset was created by assigning periods and implicit deadlines such that all 15 tasks had equal utilisation. The periods were generated by multiplying the execution times by a constant c such that $T_i = c\ C_i$ for all tasks. We varied c from 15 upwards in steps of 0.25, which varied the utilisation from 1.0 downwards. In order to evaluate different approaches, we found the *breakdown utilisation* [17] of the tasksets. By scaling the deadlines and periods of the tasks, we simulated scaling the speed of the CPU and memory. Using this technique the breakdown utilisation, the point at which the taskset is deemed unschedulable, can be found.

The breakdown utilisation for each approach is shown in Table I. The ECB-Union Multiset, and hence the combined multiset, approach performed the best with a breakdown utilisation of 0.659. The JCR approach outperformed the ECB-Only and UCB-Only approaches with a breakdown utilisation of 0.488, but did worse than the other approaches presented in this paper.

TABLE I
BREAKDOWN UTILISATION FOR THE CASE STUDY TASKSET FOR THE DIFFERENT APPROACHES USED TO CALCULATE THE CRPD

	Breakdown utilisation
No pre-emption cost	1
Combined Multiset	0.659
ECB-Union Multiset	0.659
UCB-Union Multiset	0.594
ECB-Union	0.612
UCB-Union	0.583
UCB-Only	0.462
ECB-Only	0.364
JCR	0.488

II. Experimental Evaluation

In addition to the case study, we evaluated the schedulability tests for EDF with integrated CRPD analysis using synthetically generated tasksets. This enabled us to investigate the behaviour of the different approaches as we varied a number of key parameters.

The UUnifast algorithm [10] was used to calculate the utilisation, U_i of each task so that the utilisations add up to the desired utilisation level for the taskset. Task periods $T_{i,}$ were generated at random between 5ms and 500ms according to a log-uniform distribution. From this, C_i was calculated via $C_i = U_i\ T_i$.

We generated two sets of tasksets, one with implicit deadlines and one with constrained deadlines. We used $D_i = \min(T_i,\ 2C_i + x(T_i - 2C_i))$ to generate the constrained deadlines, where x is a random number between 0 and 1. In the following section we present the results for implicit

[1] http://www.mrtc.mdh.se/projects/wcet/benchmarks.html

[2] http://www.arm.com/products/processors/classic/arm7/index.php

deadline tasksets. Results for these experiments for tasksets with constrained deadlines are available in Appendix 1 of the technical report [20] on which this paper is based. In general, using constrained deadlines resulted in an overall reduction in schedulable tasksets compared to implicit deadline tasksets.

The UCB percentage for each task was based on a random number between 0 and a maximum UCB percentage specified for the experiment. UCBs were placed in a continuous group at the start of the task's ECBs.

A. *Baseline Experiments*

A number of experiments were run in order to investigate how the integrated CRPD and EDF schedulability analysis performed under varied cache and task configurations. These experiments varied the following parameters:

- Cache utilisation (default of 10)
- Maximum UCB percentage (default of 30%)
- Number of tasks (default of 10)
- Number of cache sets (default of 256)
- Block Reload Time (BRT) (default of 8μs)

Here, cache utilisation describes the ratio of the total size of the tasks to the size of the cache. A cache utilisation of 1 means that the tasks fit exactly in the cache, whereas a cache utilisation of 5 means the total size of the tasks is 5 times the size of the cache. We used 10,000 tasksets per experiment, and assumed a direct mapped cache.

The first experiment investigates how the integrated CRPD and EDF schedulability analysis performed under the default configuration for implicit deadline tasksets. We varied the utilisation, excluding any pre-emption cost, from 0.025 to 1 in steps of 0.025 and recorded how many tasksets were deemed schedulable by the EDF schedulability test. The results are shown in Fig. 6 and in Table II in the form of weighted schedulability measures, see next sub-section, sub-section B for a definition.

The results follow a similar pattern to the equivalent CRPD analyses for FP scheduling, see Figure 9 in [3]. Furthermore, the results confirm the dominance relationships between approaches with the combined multiset approach performing the best. Additionally, with the exception of ECB-Only, all of the approaches presented in this paper outperformed JCR with the combined multiset approach achieving a weighted schedulability measure of 0.528 compared to 0.333 for JCR.

The second experiment was effectively a repeat of the first, but with constrained deadlines and it showed an overall reduction in the number of schedulable tasksets due to the tighter deadlines. However, the JCR approach performs better than with implicit deadlines, outperforming ECB-Only and UCB-Only. This is because the number of times task τ_j pre-empts task τ_k, $P_j(D_k)$, is reduced. (As D_k is now smaller than T_k, and smaller in relation to T_j, there is a smaller window in which task τ_j can pre-empt task τ_k). A graph showing the results for this experiment is available in [20].

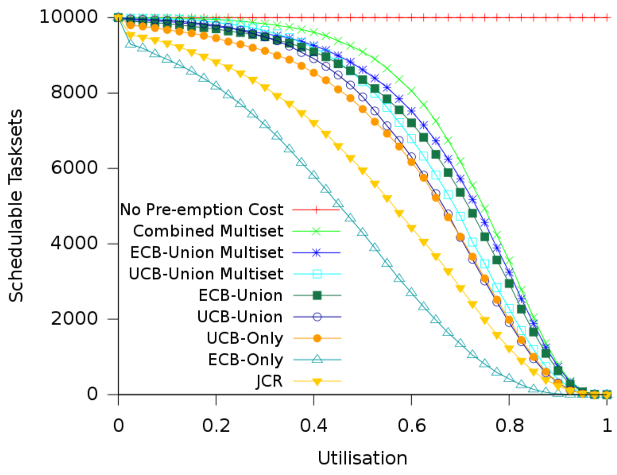

Fig. 6 – Schedulable tasksets vs Utilisation for the baseline parameters under implicit deadlines

B. *Weighted Schedulability*

Evaluating all combinations of different parameters is not possible. Therefore, the majority of our experiments focused on varying one parameter at a time. To present the results, weighted schedulability measures [7] are used. This allows a graph to be drawn which shows the weighted schedulability, $W_l(p)$, for each method used to obtain a layout l as a function of parameter p. For each value of p, this measure combines the data for all of the generated tasksets τ for all of a set of equally spaced utilisation levels, where the utilisation is based on no pre-emption cost. The schedulability test returns a binary result of 1 or 0 for each layout at each utilisation level. If this result is given by $S_l(\tau,p)$, and $u(\tau)$ is the utilisation of taskset τ, then:

$$W_l(p) = \left(\sum_{\forall \tau} u(\tau) \bullet S_l(\tau, p) \right) \Big/ \sum_{\forall T} u(\tau) \tag{34}$$

The benefit of using a weighted schedulability measure is that it reduces a 3-dimensional plot to 2 dimensions. Individual results are weighted by taskset utilisation to reflect the higher value placed on a being able to schedule higher utilisation tasksets.

Table II gives the weighted schedulability measures for the baseline experiment i.e. for each line shown on Figure 6.

TABLE II
WEIGHTED SCHEDULABILITY MEASURES FOR THE BASELINE EXPERIMENTS SHOW IN FIGURE 6

	Weighted schedulability
No pre-emption cost	1
Combined Multiset	0.528
ECB-Union Multiset	0.501
UCB-Union Multiset	0.455
ECB-Union	0.481
UCB-Union	0.427
UCB-Only	0.416
ECB-Only	0.236
JCR	0.333

C. Weighted Schedulability Experiments

For these weighted schedulability experiments, we used 1,000 tasksets, rather than 10,000 tasksets. The following results are all for implicit deadline tasksets.

Cache Utilisation and Maximum UCB Percentage

As the cache utilisation increases, see Fig. 7, all approaches that consider CRPD show a decrease in schedulability. In particular, the ECB-Only approach shows a very rapid decrease because the cache utilisation directly correlates with the number of ECBs which is all that the approach considers. Additionally, the JCR approach starts to drop off at around a cache utilisation of 8, and by a cache utilisation of 14, it performs the worst. This is due to the pessimistic handling of nested pre-emptions leading to it calculating that the same UCBs are evicted multiple times as tasks share an increasing number of cache blocks.
As the maximum UCB percentage increases, see Fig. 8, all approaches except ECB-Only show a decrease in schedulability. The ECB-Only approach shows no change because it does not consider any tasks' UCBs. The UCB-Only approach is particularly vulnerable to high numbers of UCBs. Additionally, the JCR approach also shows a large decrease in the number of schedulable tasksets. This is because it deals with nested pre-emptions by considering the pre-empting and intermediate tasks individually. As the number of UCBs increases, the chances of the analysis assuming that the UCBs get evicted more than once increases.

Number of Tasks and Cache Size

As the number of tasks increases, see Fig. 9, all approaches that consider pre-emption cost show a decrease in schedulability due to the increased number of pre-emptions. We note that as the number of tasks becomes very high, some of the approaches level off. This is due to the fact that the other parameters, specifically cache utilisation and maximum UCB percentage are fixed. As the number of tasks increases, the size of the tasks, and therefore the number of UCBs decreases, reducing the cost of a pre-emption, especially for the approaches that rely heavily on the number of UCBs. This could be avoided by fixing the task size by increasing the cache utilisation, but then this in turn would affect the results.

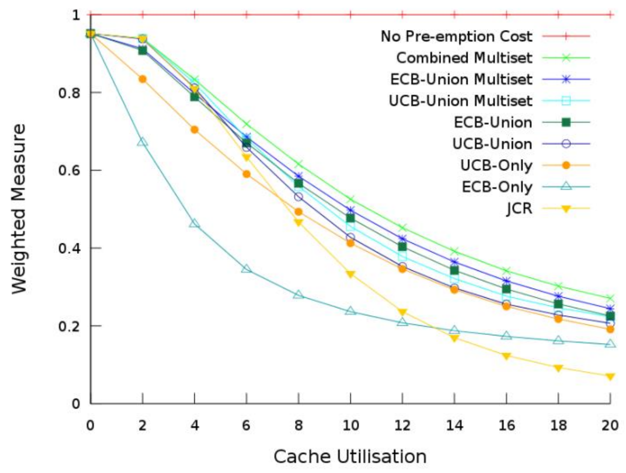

Fig. 7 – Weighted schedulability measure; varying cache utilisation from 0 to 20 in steps of 2

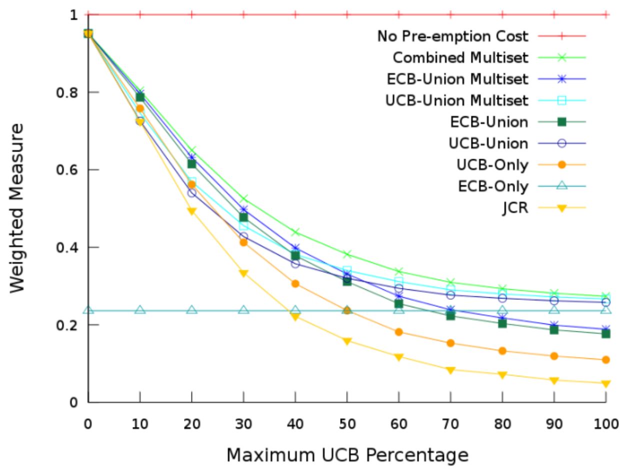

Fig. 8 – Weighted schedulability measure; varying the maximum UCB percentage from 0 to 100% in steps of 10%

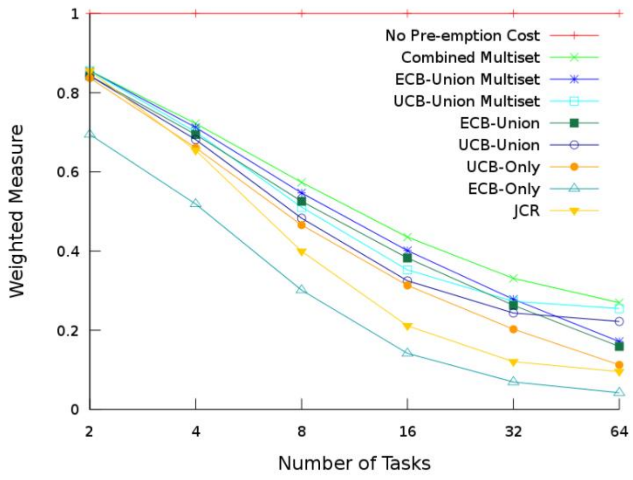

Fig. 9 – Weighted schedulability measure; varying the number of tasks from $2^1 = 2$ to $2^6 = 64$ for implicit deadline tasksets

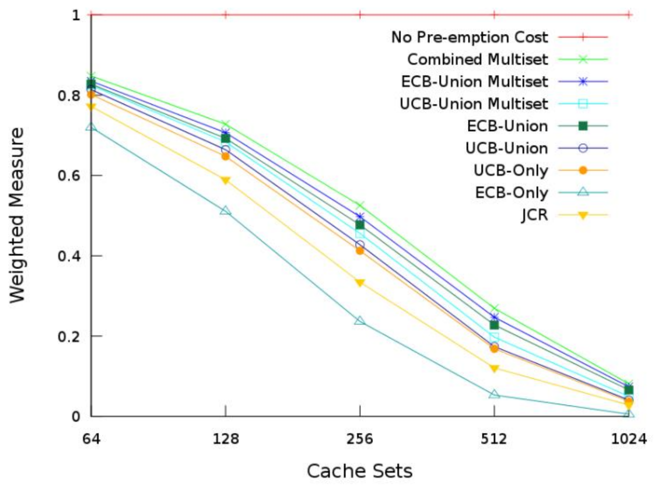

Fig. 10 – Weighted schedulability measure; varying the number of cache sets from $2^6 = 64$ to $2^{10} = 1024$ for implicit deadline tasksets

The cache size also has an effect of the schedulability of tasksets, see Fig. 10. As the number of cache sets increases, all approaches show a decrease in schedulability because the potential impact of a pre-emption increases. Varying the BRT also has a similar effect of increasing the cost of a pre-emption which in turn results in fewer tasksets being deemed schedulable. A graph for this experiment is available in [20].

III. Conclusion

In this paper, we have presented new CRPD aware analysis for the EDF scheduling algorithm based on similar work for FP scheduling. We compared our new approaches against an existing approach for EDF by Ju *et al.* [15], referred to as JCR, and showed that our combined multiset approach dominates the JCR approach. This was confirmed in both a case study and experiments based on synthetically generated tasksets. Through a series of experiments, we examined the effects of different cache and taskset parameters on the different approaches, highlighting the strengths and weaknesses of the different approaches. We found that the JCR approach was especially vulnerable to high numbers of tasks, high cache utilisation and high UCB percentages. In all of our experiments, our new combined multiset approach was able to schedule the highest number of tasksets out of the approaches that consider CRPD.

In the future, we aim to perform a detailed comparison between FP and EDF scheduling with integrated CRPD analysis. Additionally, we aim to perform a more comprehensive case study using code from a multitasking application.

Acknowledgements

This work was partially funded by the UK EPSRC through the Engineering Doctorate Centre in Large-Scale Complex IT Systems (EP/F501374/1), the UK EPSRC funded Tempo project (EP/G055548/1), the Transregional Collaborative Research Center AVACS of the German Research Council (DFG), and the European Community's ARTEMIS Programme and UK Technology Strategy Board, under ARTEMIS grant agreement 295371-2 CRAFTERS. The authors would like to thank Antoine Colin for his incisive comments on a previous draft of this paper.

References

[1] S. Altmeyer and C. Burguière, "Cache-related Preemption Delay via Useful Cache Blocks: Survey and Redefinition," *Journal of Systems Architecture*, 2010.

[2] S. Altmeyer, R.I. Davis, and C. Maiza, "Cache Related Pre-emption Delay Aware Response Time Analysis for Fixed Priority Pre-emptive Systems," in *Proceedings of the 32nd IEEE Real-Time Systems Symposium (RTSS)*, Vienna, Austria, 2011, pp. 261-271.

[3] S. Altmeyer, R.I. Davis, and C. Maiza, "Improved Cache Related Pre-emption Delay Aware Response Time Analysis for Fixed Priority Pre-emptive Systems," *Real-Time Systems*, vol. 48, no. 5, pp. 499-512, September 2012.

[4] S. Baruah and A. Burns, "Sustainable Scheduling Analysis," in *Proceedings of the 27th IEEE Real-Time Systems Symposium (RTSS)*, 2006, pp. 159-168.

[5] S. K. Baruah, A. K. Mok, and L. E. Rosier, "Preemptive Scheduling Hard-Real-Time Sporadic Tasks on One Processor," in *Proceedings of the 11th IEEE Real-Time Systems Symposium (RTSS)*, Lake Buena Vista, Florida, USA, 1990, pp. 182-190.

[6] S. K. Baruah, L. E. Rosier, and R. R. Howell, "Algorithms and Complexity Concerning the Preemptive Scheduling of Periodic Real-Time Tasks on One Processor," *Real-Time Systems*, vol. 2, no. 4, pp. 301-324, 1990.

[7] A. Bastoni, B. Brandenburg, and J. Anderson, "Cache-Related Preemption and Migration Delays: Empirical Approximation and Impact on Schedulability," in *Proceedings of OSPERT*, Brussels, Belgum, 2010, pp. 33-44.

[8] M. Bertogna and S. Baruah, "Limited Preemption EDF Scheduling of Sporadic Task Systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 4, pp. 579-591, November 2010.

[9] M. Bertogna, O. Xhani, M. Marinoni, F. Esposito, and G. Buttazzo, "Optimal Selection of Preemption Points to Minimize Preemption Overhead," in *Proceedings of 23rd Euromicro Conference on Real-Time Systems (ECRTS)*, Porto, Portugal, 2011, pp. 217-227.

[10] E. Bini and G. Buttazzo, "Measuring the Performance of Schedulability Tests ," *Real-Time Systems*, vol. 30, no. 1, pp. 129-154, 2005.

[11] J. V. Busquets-Mataix, J. J. Serrano, R. Ors, P. Gil, and A. Wellings, "Adding Instruction Cache Effect to Schedulability Analysis of Preemptive Real-Time Systems," in *Proceedings of the 2nd IEEE Real-Time Technology and Applications Symposium (RTAS)*, 1996, pp. 204-212.

[12] M. L. Dertouzos, "Control Robotics: The Procedural Control of Physical Processes," in *Proceedings of the International Federation for Information Processing (IFIP) Congress*, 1974, pp. 807-813.

[13] L. George, N. Rivierre, and M. Spuri, "Preemptive and Non-Preemptive Real0Time Uniprocessor Scheduling," INRIA, Technical Report 2966, 1996.

[14] J. Gustafsson, A. Betts, A. Ermedah, and B. Lisper, "The Mälardalen WCET benchmarks – past, present and future," in *Proceedings of the 10th International Workshop on Worst-Case Execution Time Analysis (WCET'2010)*, Brussels, Belgium, September 2010, pp. 137-147.

[15] l. Ju, S. Chakraborty, and A. Roychoudhury, "Accounting for Cache-Related Preemption Delay in Dynamic Priority Schedulability Analysis," in *Design, Automation and Test in Europe Conference and Exposition (DATE)*, Nice, France, 2007, pp. 1623-1628.

[16] C. Lee et al., "Analysis of Cache-related Preemption Delay in Fixed-priority Preemptive Scheduling," *IEEE Transactions on Computers*, vol. 47, no. 6, pp. 700-713, June 1998.

[17] J. Lehoczky, "The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior," in *Proceedings of the 10th Real Time Systems Symposium (RTSS)*, Santa Monica, California, USA, 1989, pp. 166-171.

[18] J. Y.-T. Leung and M. L. Merrill, "A Note on Preemptive Scheduling of Periodic, Real-Time Tasks," *Information Processing Letters*, vol. 11, no. 3, pp. 115-118, 1980.

[19] C. L. Liu and J. W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment," *Journal of the ACM*, vol. 20, no. 1, pp. 46-61, January 1973.

[20] W. Lunniss, S. Altmeyer, C. Maiza, and R. I. Davis, "Intergrating Cache Related Pre-emption Delay Analysis into EDF Scheduling," University of York, York, UK, Technical Report YCS-2012-478. Available from http://www-users.cs.york.ac.uk/~wlunniss/, 2012.

[21] I. Ripoll, A. Crespo, and A. K. Mok, "Improvement in Feasibility Testing for Real-Time Tasks," *Real-Time Systems*, vol. 11, no. 1, pp. 19-39, 1996.

[22] M. Spuri, "Analysis of Deadline Schedule Real-Time Systems," INRIA, Technical Report 2772, 1996.

[23] Y. Tan and V. Mooney, "Timing Analysis for Preemptive Multitasking Real-Time Systems with Caches," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 1, February 2007.

[24] F. Zhang and A. Burns, "Schedulability Analysis for Real-Time Systems with EDF Scheduling," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1250-1258, September 2009.

Appendix 1

This appendix contains the results for the experiments performed in Section VII that were omitted due to space constraints.

A. *Implicit Deadline Tasksets*

This section shows the graphs for the experiments presented and discussed in Section VII for implicit deadline tasksets.

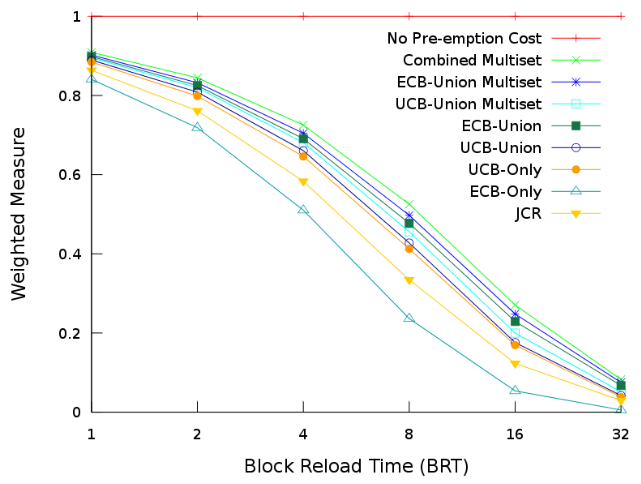

Fig. 11 – Weighted schedulability measure; varying the block reload time from $2^0 = 1\mu s$ to $2^5 = 32\mu s$ for implicit deadline tasksets

B. *Constrained Deadline Tasksets*

The following section shows the results for the experiments presented in section VII for constrained deadline tasksets. In general, using constrained deadlines resulted in an overall reduction in the number of schedulable tasksets compared to implicit deadline tasksets. However, we note that the JCR approach shows an improvement compared to the implicit deadline case for the reason noted in Section VII. Nevertheless, while it does better than the ECB-Only and UCB-Only approach, the JCR approach is still outperformed by the other approaches presented in this paper in almost all cases. Furthermore, the combined multiset approach presented in this paper always outperforms the JCR approach.

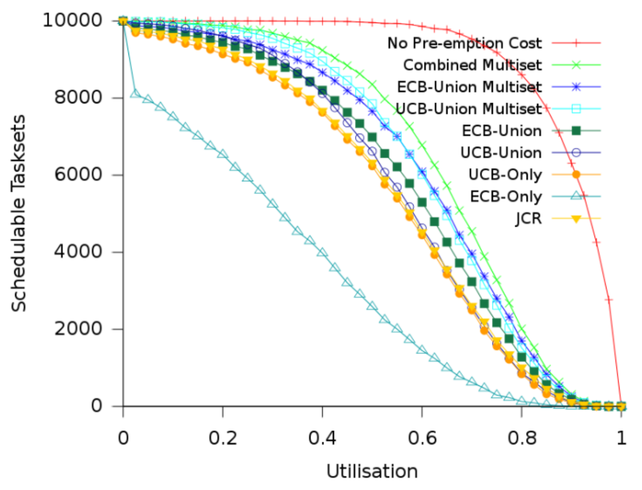

Fig. 12 – Schedulable tasksets vs Utilisation for the baseline parameters under constrained deadlines

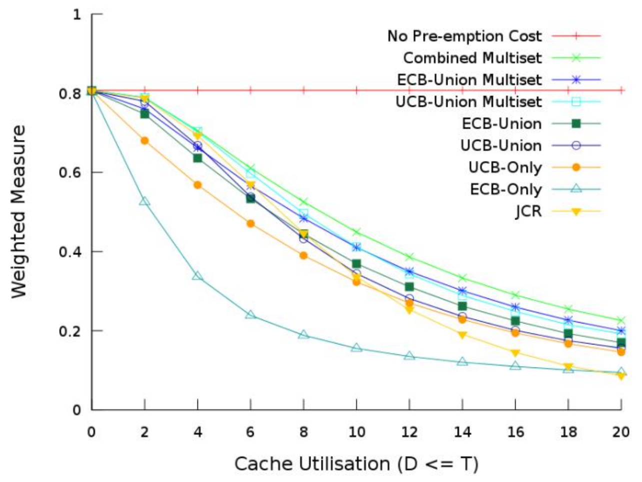

Fig. 13 – Weighted schedulability measure; varying cache utilisation from 0 to 20 in steps of 2 for constrained deadline tasksets

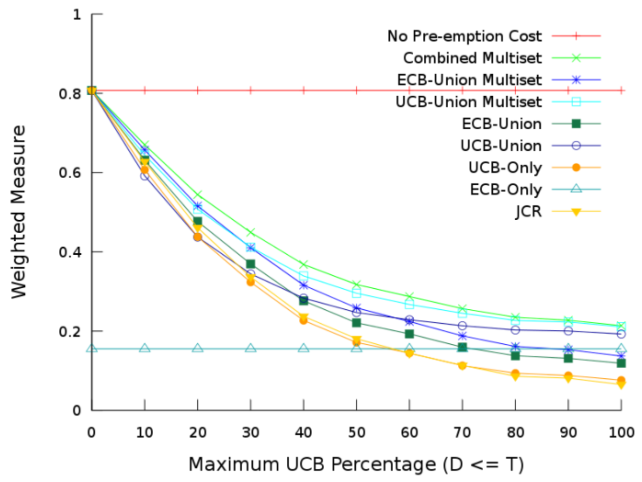

Fig. 14– Weighted schedulability measure; varying the maximum UCB percentage from 0 to 100% in steps of 10% for constrained deadline tasksets

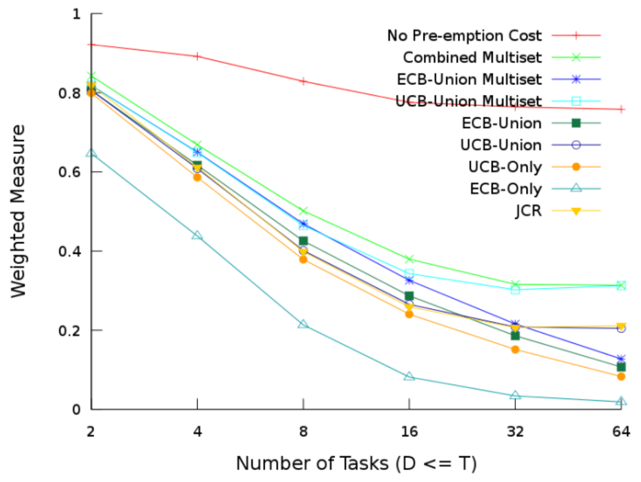

Fig. 15 – Weighted schedulability measure; varying the number of tasks from $2^1 = 2$ to $2^6 = 64$ for constrained deadline tasksets

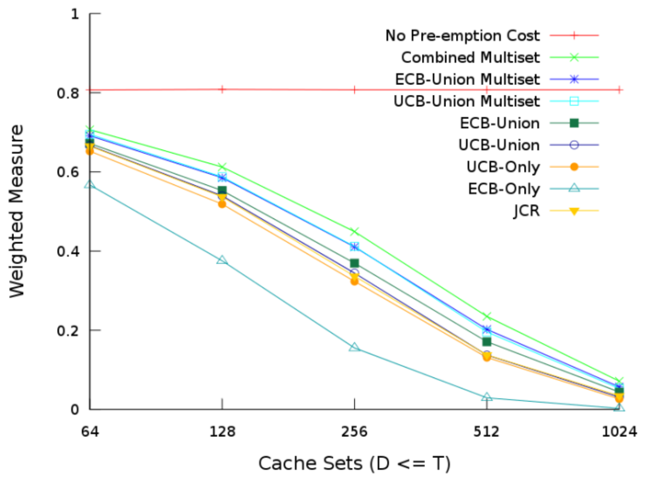

Fig. 16 – Weighted schedulability measure; varying the number of cache sets from $2^6 = 64$ to $2^{10} = 1024$ for constrained deadline tasksets

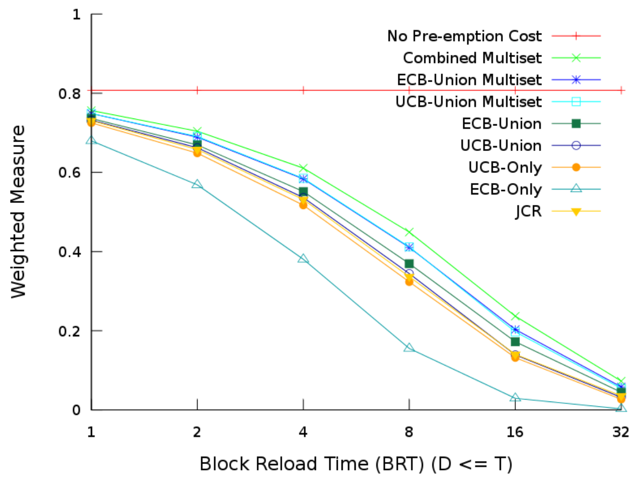

Fig. 17 – Weighted schedulability measure; varying the block reload time from $2^0 = 1\mu s$ to $2^5 = 32\mu s$ for constrained deadline tasksets