Preemptive real-time schedulers are associated with preemption related overheads, and their effects are challenging to analyze because they typically vary with the point of preemption e.g., cache related preemption delays, and even with the state of the physical process that the real-time system is controlling. Moreover, preemptive scheduling typically requires the use of resource access protocols [1] to enable mutual exclusion, in cases where tasks communicate through shared resources. These resource access protocols, though predictable, introduce schedulability overheads in the system, as well as may lead to pessimistic assumptions in the schedulability analysis. Even though preemptive scheduling schemes are used in a large number of applications, mostly due to its ability to achieve high processor utilization, the detrimental impact of preemptions is widely recognized in the community [2] [3] [4]. The preemption related costs includes the context-switch overhead [5] and to manipulate the task queues [3], as well as the indirect cost of cache-related preemption delays [6]. Bui et al [2] observed a worst case increment in task execution time of up to 33% on a PowerPC MPC7410 with a 2 MB two way associative L2 cache, due to the cache related preemption delays. The worst case temporal overhead due to cache related preemption delays were found to be as high as 655µS for a single preemption. The rate monotonic algorithm (RM) was shown to introduce a higher number of preemptions than earliest deadline first algorithm (EDF) by Buttazzo [7], increasing the overheads in the system and hence reducing the benefits of its simple runtime implementation.

The applicability of a non-preemptive scheduling scheme, on the other hand, is limited to only a small fraction of the feasible task sets [8] due to its inability to fully utilize the computational resources in most of the cases [9]. Task sets scheduled by non-preemptive schemes can be deemed as unschedulable even at arbitrarily low utilizations [10]. On the other hand, the major benefits of using non-preemptive scheduling is the absence of preemption related costs and that it does not require any resource access protocols, as mutual exclusion is guaranteed by the scheduler. However, the problem of finding a non-preemptive schedule for a given task set is either NP-hard or infeasible for most task sets [11].

In order to take advantage of the benefits of both preemptive and non-preemptive scheduling paradigms, various limited preemption scheduling models were proposed, a detailed survey of which can be found in [12]. A benefit of using these approaches is that non-preemptive regions can be enforced within the task executions. The use of a limited preemption technique helps in bounding the preemption related costs and reducing it, i.e., it basically ensures that a preemption occurs only when absolutely necessary and/or at an optimal point with the least cost. It also ensures that, whenever possible, critical sections can be executed entirely non-preemptively. However, the limited preemption approach does not provide for a fine grained ability to control the preemption behavior of real-time tasks e.g., to guarantee a user specified bound on the number of preemptions per task. To guarantee mutual exclusion during critical section execution, it is essential that the non-preemptive region of each task is larger than its largest critical section. Similarly, to minimize preemption related costs, the length of the non-preemptive region of any task must be no less than the length of the task execution between any two consecutive optimal preemption points, where the cost of a preemption is the least. If the length of the non-preemptive region does not guarantee a specified preemption behavior, a preemptive or non-preemptive schedule may not be feasible. For example, if the preemptions are not possible at optimal preemption points, it may increase the task execution time by 33% [2], potentially causing deadline misses. Augmenting the limited preemption scheduling schemes with the flexibility for specifying a specified non-preemption behavior can further enhance its applicability in modern real-time systems to control the preemption related costs.

Of the several methods that have been proposed to reduce the number of preemptions in real-time scheduling, Preemption Threshold Scheduling (PTS) for FPS was first introduced in the ThreadX operating system by Lamie [13]. This scheme was later formalized by [14], by providing an accurate analysis and also proposing an optimal algorithm to calculate the preemption thresholds. Jeffay et al [11] derived the sufficient and necessary conditions for non-preemptive feasibility of periodic and sporadic tasks. They also showed that EDF is an optimal algorithm for scheduling tasks non-preemptively under a work conserving paradigm. Baruah [9] proposed the limited preemption model, which was subsequently named as Floating Non-Preemptive Region model (f-NPR model), in which they proposed an algorithm to calculate the length of the longest possible non-preemptive execution of a task in a sporadic task system. Later, [15] evaluated the approach for
Consequently, our goal is to find the processor speed, S, that satisfies:

\[ q_{i,j}^S \leq L_i^S = \max_{1 \leq j < m} (q_{i,j+1}^S - q_{i,j}^S, q_{i,1}^S) \]

Consequently, our goal is to find the processor speed, S, that satisfies:

\[ Q_i^S \geq L_i^S = \max_{1 \leq j < m} (q_{i,j+1}^S - q_{i,j}^S, q_{i,1}^S) \]
Thus if the processor executes at speed $S$, any preemption on $\tau_i$ can be deferred to the closest optimal preemption point.

An illustrative example is given in figure 1 where the non-preemption requirement of a task that guarantees preemptions only at optimal preemption points, is greater than its largest non-preemptive execution. In the figure, the actual length of the largest non-preemptive regions is calculated using the result by [9]. If a high priority task is released immediately after the optimal preemption point $op_1$, the preemption cannot be deferred to the next optimal preemption point $op_2$ as the largest possible non-preemptive region for $\tau_i$ is less than the length of the task execution between $op_1$ and $op_2$. In our example, for $\tau_i$, the largest length of the task execution between any two optimal preemption point is given by the length of the task execution between $op_1$ and $op_2$. Hence, to ensure that the preemptions can always occur at one of the optimal preemption point, the length of the largest possible non-preemptive region should be at least as large as the length of the task execution between $op_1$ and $op_2$. Only then can we ensure that any preemption that may potentially occur after $op_1$ can be guaranteed to be deferred to the optimal preemption point $op_2$.

C. Execute Critical Sections within Non-Preemptive Regions

An attractive feature of the limited preemption scheduling paradigm is that it has the potential to enable access to shared resources without the need for synchronization mechanisms, by executing the critical sections within non-preemptive regions. However, this is not always possible, e.g., in case the length of the non-preemptive region $Q_i^S$, for a task $\tau_i$ is shorter than its largest critical section $CS_i^S$ on a processor of speed $S$. This issue, on the other hand, can be solved by finding an adequate processor speed $S$ that enables $Q_i^S \geq L_i^S$. The processor speed that guarantees this is given by the speed $S$ that will satisfy the relation:

$$Q_i^S \geq L_i^S = CS_i^S$$

An illustrative example is given by figure 2 where the non-preemption requirement of a task that guarantees the non-preemptive execution of its critical sections is greater the actual length of its largest non-preemptive region.
III. RELAXING THE ASSUMPTION OF LINEAR SPEED-UP

In this section, we discuss how we can relax some of the assumptions made in this paper. Specifically, we consider relaxing the assumptions of linear speed-up and negligible preemption related overheads at optimal preemption points.

So far in this paper, we have considered the possibility of a linear speed-up. While such an assumption allows us to derive interesting theoretical results on the resource augmentation bounds for preemption control, it might not be a reasonable assumption from a practical point of view because of the effects of memory wall [26]. In this sub-section, we consider relaxing the assumption of a linear speed-up by considering the execution time model proposed by [27]. Marinoni et al [27] assumed that the execution time of a task consists of two parts- one that scales linearly with the processor frequency and one that does not scale with the processor frequency. Following their notation, let us define \( \phi \) as the percentage of the execution time for every \( \tau_i \), that scales with the processor frequency.

Let us see how the theorem IV.1 changes when we consider our new assumption.

**Theorem III.1.** The processor speed \( S_i \) that guarantees the feasibility of a non-preemption requirement \( L_i \) for a task \( \tau_i \) is given by,

\[
S_i = \max_{D_1 \leq t < D_i} \left\{ \frac{\phi \sum_{j=1}^{n} DBF_j^1(t)}{t - L_i - (1 - \phi) \sum_{j=1}^{n} DBF_j^1(t)} \right\}
\]

*Proof:* The length of the non-preemptive region for \( \tau_i \) at speed 1 is given by [9],

\[
Q_i^1 = \min_{D_1 \leq t < D_i} \left\{ t - \sum_{j=1}^{n} DBF_j^1(t) \right\}
\]

\[
\Rightarrow Q_i^1 \leq t - \sum_{j=1}^{n} DBF_j^1(t), \forall t, D_1 \leq t < D_i
\]

Our aim is to find the processor speed \( S_i \) that guarantees the feasibility of a non-preemption requirement \( L_i \). We know that, of the total demand bound in any interval \( t \), only \( \phi \) percentage scales with the processor frequency. Thus,

\[
L_i \leq t - \left\{ \frac{\phi \sum_{j=1}^{n} DBF_j^1(t)}{S_i} + (1 - \phi) \sum_{j=1}^{n} DBF_j^1(t) \right\}, \forall t, D_1 \leq t < D_i
\]

Hence,

\[
S_i L_i \leq S_i t - \left\{ \phi \sum_{j=1}^{n} DBF_j^1(t) + S_i (1 - \phi) \sum_{j=1}^{n} DBF_j^1(t) \right\}, \forall t, D_1 \leq t < D_i
\]

Solving for \( S_i \), we get,

\[
S_i \geq \left\{ \frac{\phi \sum_{j=1}^{n} DBF_j^1(t)}{t - L_i - (1 - \phi) \sum_{j=1}^{n} DBF_j^1(t)} \right\}, \forall t, D_1 \leq t < D_i
\]

i.e.,

\[
S_i = \max_{D_1 \leq t < D_i} \left\{ \frac{\phi \sum_{j=1}^{n} DBF_j^1(t)}{t - L_i - (1 - \phi) \sum_{j=1}^{n} DBF_j^1(t)} \right\}
\]

**Lemma III.1.** The speed \( S_i \) that guarantees the feasibility of a non-preemption requirement \( L_i \) for any task \( \tau_i \), during a time interval \( t \) is upper-bounded by,

\[
S_i \leq \frac{\phi y}{\phi y - 1}
\]

where, \( y = \frac{t}{L_i} \), \( \forall t \in [D_1, D_i) \).

*Proof:* We know from theorem III.1 that,

\[
S_i = \max_{D_1 \leq t < D_i} \left\{ \frac{\phi \sum_{j=1}^{n} DBF_j^1(t)}{t - L_i - (1 - \phi) \sum_{j=1}^{n} DBF_j^1(t)} \right\}
\]
Since we have assumed that the task set is feasible, the upper-bound on the value of \( \sum_{j=1}^{n} DBF_j(t) \) is \( t \). Hence,

\[
S_i \leq \left\{ \frac{\phi t}{t - L_i - (1 - \phi) t} \right\}
\]

which gives,

\[
S_i \leq \frac{t}{\phi t - L_i}
\]

Finally, substituting \( y = \frac{t}{L_i} \),

\[
S_i \leq \frac{\phi y}{\phi y - 1}
\]

We can now use similar reasoning as in lemma IV.2, IV.3 and IV.4 to derive upper-bounds on the processor speed-up required that provides the required non-preemption guarantees. We however leave more details to a future work, since in this paper we focus on the fundamental question of how the preemption behavior changes with the processor speed.

REFERENCES


