Assessing the Pessimism of Current Multicore Global Fixed-Priority Schedulability Analysis

Youcheng Sun, Marco Di Natale

Assessing the Pessimism of Current Multicore Global Fixed-Priority Schedulability Analysis

Youcheng Sun
University of Oxford, UK
youcheng.sun@cs.ox.ac.uk

Marco Di Natale
Scuola Superiore Sant'Anna, Pisa, Italy
marco@sssup.it

Abstract

This work provides a formal assessment on the pessimism of existing methods for the schedulability analysis of multicores with global fixed priority (FP) scheduling. We show how according to the existing analysis methods for FP scheduling, it is relatively easy to define a simple task allocation strategy followed by local analysis that dominates the existing global-FP feasibility analysis algorithms, in terms of deadline guarantees. Rather than being an indication of a true comparison between the effectiveness of local and global policies, we consider the result as an indication of the limited maturity of multicore global analysis (and its outstanding pessimism).

Additionally, we show how a simple change in the task model, consisting in splitting the task execution and performing the analysis in two stages, allows to provide a better global schedulability analysis that overcomes this limitation. However, the new analysis mainly aims to show where the demonstrated pessimism most likely comes from. Due to the huge pessimism pointed out on existing analyses and the limited improvement from the new analysis, more efforts from the community are needed to enhance the multicore global scheduling works.

1 Introduction and State of the Art

A real-time task is a piece of software codes that is repeatedly activated with specified (minimum) inter-arrival time. Each task instance can be called a *job*, which is further characterised by its arrival time and an absolute deadline such that a job must complete its execution in between the two time instants: the duration of a job's execution is called its response time. A task is said to be schedulable if all its jobs can meet the respective deadlines.

The real-time scheduling problem deals with how to schedule a set of tasks so as to guarantee that they complete execution before their deadlines on a single-core or multicore platform based on simple assumptions on their (worst-case) execution times. The problem consists of two sides: defining an effective scheduling policy and finding an analysis method that allows to provide the required guarantee for the selected policy (or a class of them).

This work targets the very common model of periodic or sporadic independent tasks that need to be scheduled on a set of cores. Each task is further characterized by its worst-case execution time and relative deadline.

Since the seminal work of Liu and Layland (1973), the Fixed Priority (FP) and the Earliest Deadline First (EDF) scheduling algorithms have been overwhelmingly popular. According to the FP scheduling, a task is statically assigned a priority, and all of its jobs execute according to it. At runtime, the arrival of a higher priority job can preempt the execution of a lower priority job. In EDF scheduling, the priority is assigned to each job separately and depends on its absolute deadline, such that an earlier deadline corresponds to a higher priority. Both FP and EDF belong to the class of job-level fixed-priority scheduling policies.

In the case of the single core, most existing analysis results make use of the critical instant theorem, which provides knowledge on the worst-case combination of task arrival times that results in the worst-case response time for a given task. As shown in Liu and Layland (1973), the worst-case scenario for single core analysis is when all tasks are simultaneously activated and all tasks release their jobs as soon as possible. By analyzing this job release pattern, the exact schedulability analysis upon the single core is tractable (in pseudo-polynomial time) for both FP (Audsley et al (1991)) and EDF scheduling (Baruah et al (1990); Spuri (1996)).

Upon multicores, the available scheduling policies can be classified as *partitioned* or *global*.

In partitioned scheduling, tasks are statically allocated to cores, and the tasks associated to a core are scheduled as in the single core case. The task allocation problem for the partitioned scheduling can be formulated as a bin packing problem (Baruah and Bini (2008)). Even if not optimal, there are heuristic algorithms like those in Fisher et al (2006) and Chen (2015) that can find efficient solutions to the partitioning problem.

In global scheduling, a task is allowed to freely migrate among the available cores (typically with a simplified assumption of zero migration cost). A preemption can happen if a higher priority job arrives but there is no idle core. In this case, the preeempted task is put back in a global ready queue and may resume its execution upon the first core that becomes available (the same where it was executing or a different one). In correspondence with the FP algorithm and the EDF algorithm, global scheduling can be performed using Global FP (G-FP) or Global EDF (G-EDF) upon the multicore.

Partitioned scheduling and global scheduling are incomparable. For example, if a task is schedulable by G-FP (G-EDF) scheduling, then it may or may not be schedulable by partitioned FP (EDF) scheduling, and vice versa. More

information for multicore real-time scheduling is surveyed in Davis and Burns (2011b).

In this work, we focus on multicore global scheduling. Different from the single core case, in the analysis of multicore global scheduling, the critical instant condition cannot be easily identified and there is no simple way to find the absolute worst case set of job arrivals. An illustrative discussion on the time critical condition can be found in Baruah (2007); Davis and Burns (2011b). Davis and Burns (2011a) and Sun et al (2014) independently identified a (infinite) class of task activations that lead to the worst-case response time of a task. In fact, there is currently no single worst-case scenario for analysing the schedulability of a task in the case of global scheduling. An exact analysis would need to consider all the possible task arrival patterns and is clearly impractical. Therefore, the state of the art for multicore analysis of global scheduling relies on a set of sufficient-only conditions that are to some degree pessimistic.

This creates two open problems: first, there is currently no simple way to compare the effectiveness of partitioned versus global scheduling, even under the assumption of availability of good (but necessarily suboptimal) partitioning schemes. The second problem is how to improve the quality of the existing analysis methods to reduce their pessimism and hopefully to achieve a measure of their overall quality in comparison to an ideal sufficient and necessary test.

Contributions

- The main result of this paper is a formal proof that demonstrates how existing multicore global scheduling analysis methods are pessimistic enough to be always dominated (only in terms of scheduling guarantees, not real performance) by a simple partitioning scheme followed by exact local analysis.
- Another minor contribution is that we show how most likely the pessimism derives from limitations in the analysis method, not the scheduling policy. It is shown that a very simple change in the input model for the analysis, consisting in considering the execution of a task as the sequential execution of two segments (this does not change the actual task model, nor its scheduling) is sufficient for an improvement of the analysis results. This contribution is partly of value as a new way of applying the existing analysis methods, since it provides better/tighter schedulability conditions, but mostly points at the need for better methods and better ways to reduce pessimism.

1.1 Related work

A set of tests have been developed for the exact analysis of globally scheduled tasks under the assumption of *discrete time*, that is, all tasks must arrive only at integer time instants. Bertogna and Cirinei (2007) proposed the first test based on explicitly modeling the schedulability analysis as a finite

state machine and computing the state space. Geeraerts et al (2013) enhanced Baker and Cirinei's solution by investigating the simulation relation to eliminate the unnecessary states that are not useful for assessing the schedulability. In Bonifaci and Marchetti-Spaccamela (2012), the multicore global schedulability problem was studied and was formulated into a two-player game and this motivated Burmyakov et al (2015) to develop a faster exact test for multicore G-FP scheduling. Due to the complexity of the problem, all these methods can only deal with task parameters with rather small integers. This constraint holds, even when considering only a task's periodic behavior. Examples of works fousing on the exact global schedulability analysis of periodic tasks upon multicores include Guan et al (2007) and Cucu-Grosjean and Goossens (2011).

As shown in Sun (2015), the multicore global scheduling in discrete time is not robust with respect to the selection of the time granularity and guarantees are not retained if a smaller step is selected (or against drifts or inaccuracies in the assumptions). For example, given a set of tasks that are schedulable by G-FP (assuming discrete time), schedulability may not be preserved if all task parameters are scaled up by a common factor (such as 10). Different from the above exact solutions, Sun and Lipari (2014) solved the G-FP schedulability analysis relying on the Linear Hybrid Automata model in *continuous time*. Further, Sun and Lipari (2015a) managed to prove that the exact analysis of a set of sporadic tasks under a G-FP scheduler can be decided within a bounded time interval.

Unfortunately, nowadays exact solutions suffer from the fact that they can only deal with very small systems, thus the major effort from the community has been on providing over-approximate analysis results. Among the large number of available results, we refer to those that are most influential and we believe most related to our work.

Baker (2003) developed a schedulability analysis method that consists of selecting a time interval (or problem window) and computing the maximum workload and interference that the task under analysis may suffer within it. Bertogna et al (2005) observed that if an interfering task's workload is too large, then the part that executes in parallel with the task under analysis should not be taken into account for bounding the maximum interference. This simple observation is often used as the main approach to reason on the parallel execution of tasks in the schedulability analysis of multicores. In Bertogna and Cirinei (2007), the Response Time Analysis (RTA) technique was applied to the schedulability analysis of multicore global scheduling. Within the problem window, the interfering tasks can be divided into two classes: i) carry-in (CI) tasks with a job that is released before the start of the window but finishes within the window, and ii) non-carry-in (NC) tasks. An interfering task can bring more interference if it has carry-in. Baruah (2007) developed the limited carry-in technique by formulating a problem window such that the number of of CI tasks within it is bounded by the number of cores minus by one. Although the technique in Baruah (2007) was originally designed for G-EDF scheduling, Guan et al (2009) combined it with the RTA in Bertogna and Cirinei (2007) and improved the schedulability analysis for G-FP scheduling. Similarly as in Guan et al (2009),

the RTA procedure and the limited carry-in technique have been integrated in Sun and Lipari (2015b) for improving the schedulability analysis of G-EDF scheduling. Lee and Shin (2013) made the attempt to apply the limited carry-in technique to the general class of global scheduling policies.

The test in Guan et al (2009) has been regarded as the state-of-the-art G-FP schedulability analysis. Sun et al (2014) demonstrated and fixed the optimism for the formulation of the workload in Guan et al (2009) in the case of arbitrary-deadline tasks (that is, a task is allowed to release a new job even before its current job finishes its execution), and developed a carry-in enumeration technique that explicitly considers every possible combination of CI tasks within the problem window. Though this analysis method suffers of exponential time complexity, Sun et al (2014) confirmed its applicability and performance improvement for rather large systems. Later, Huang and Chen (2015) also discovered the issue in Guan et al (2009), and proposed a new G-FP test for arbitrary-deadline tasks. Another G-FP test has been proposed in Liu and Anderson (2013), but its authors reported that there was an error in the workload formulation, and its performance was never re-validated. The G-FP test in Guan et al (2015) tries to estimate the CI task's interference more carefully; however, the improvement it brings may not be justified by its additional runtime complexity.

Most state-of-the-art tests for G-EDF are incomparable with each other. An empirical evaluation in Bertogna and Baruah (2011) showed that the tests in Bertogna and Cirinei (2007) and Baruah (2007) have better average performance than others. Recently, Sun and Lipari (2015b) proposed a G-EDF test that strictly dominates Bertogna and Cirinei (2007) and Baruah (2007) (if a taskset is judged to be schedulable by the latter is always found schedulable by Sun and Lipari (2015b)). However, the proposed method is incomparable with G-EDF tests like those in Baker and Baruah (2009a) and Baruah et al (2009).

To improve the accuracy of existing sufficient schedulability tests, Lee et al (2015) developed a compositional theory to apply a schedulability test after isolating an interfering task (typically with a very large load) on a dedicated core. Indeed, when there is a significantly large parallel execution between the interfering task and the one under analysis, some sufficient tests can be so pessimistic that it is better to assume that the interfering task always runs upon a dedicated core. In Lee et al (2015) the performance of this compositional theory was studied with several specific schedulability tests. A similar method has also been used in Pathan and Jonsson (2014) to improve the priority assignment scheme in Davis and Burns (2011a) for G-FP scheduling.

In Lee (2014), the execution time of the task under analysis is considered as the composition of two segments, and the interference is separately analyzed for each part. However, applicability of this solution is restricted to the context of the G-EDF test in Bertogna and Cirinei (2007).

Many of the above mentioned works (e.g., Guan et al (2009); Sun et al (2014)) assume *discrete time* and, as shown in Sun (2015) they are not robust with respect to the granularity of the time step or the violation of the discrete time assumption.

Organization of this paper In Section 2 we formally describe the system model in this paper. Section 3 introduces the state-of-the-art multicore global schedulability analysis and Section 4 emphasizes its limitation. To tackle such a limitation, the new schedulability analysis scheme is developed in Section 5. In Section 6, this new technique is compared with previous works. Section 7 concludes the paper.

2 System Model

A real-time task is characterized by a tuple $\tau_i = (C_i, D_i, T_i)$, where C_i is its Worst-Case Execution Time (WCET), D_i is the relative deadline and

- if τ_i is a *sporadic* task, T_i is the minimum time interval between two successive instances of the task;
- if τ_i is periodically activated, T_i is its *period*.

For simplicity, we assume that $C_i \leq D_i \leq T_i$. The utilization of a task is defined as $U_i = \frac{C_i}{T_i}$.

Each task instance is denoted as a *job*. A job is further defined by its arrival (release) time $r_{i,j} \in \mathbb{R}^+ \cup \{0\}$, its finish time $f_{i,j}$, and its absolute deadline $d_{i,j} = r_{i,j} + D_i$, where j represents the job index. The duration of a job's execution, i.e., $f_{i,j} - r_{i,j}$, is the job response time: the Worst-Case Response Time (WCRT) R_i of a task is the maximum response time among all its jobs. A task is said to be *schedulable* if its jobs always complete before their respective deadlines, that is, the task WCRT does not exceed its D_i.

We discuss the scheduling problem in the *continuous* time domain, meaning that the job arrival time $r_{i,j}$ and finish time $f_{i,j}$ are non negative real values.

A real-time system consists of a set $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ of n real-time tasks running upon m ($m < n$) identical cores. Tasks are scheduled by the *Global* Fixed Priority (G-FP) scheduling policy. By convention, we assume that a task with lower index has a higher priority, that is, if $i < k$, then τ_i has higher priority than τ_k. A system is said to be schedulable if every task is schedulable.

It is demonstrated in Baker and Baruah (2009b) that the G-FP scheduling of sporadic real-time tasks is sustainable with respect to the execution time: by decreasing the execution time of tasks in a schedulable system, the resulting system is still schedulable. Sustainability also holds for periodic tasks (the periodic activation is a special instance of the more general general case). Thus, we simply assume that every task activation requires exactly its WCET.

3 Multicore Global Schedulability Analysis

In this section, we introduce the state-of-the-art schedulability analysis methods for multicore global scheduling, and then we provide evidence of their pessimism.

We focus on the schedulability of a task τ_k, also defined as *target* task. More specifically, we analyze one arbitrary job of τ_k, or *target* job. The execution of the target task is interfered by higher priority tasks.

Multicore global schedulability analysis is based on the concept of *problem window*. A problem window is a time interval $[a, b)$ such that a is the arrival time of the target job, b is a generic value $\leq a + D_k$ and $x = b - a$ is the length of the window. The target job is schedulable if it can complete its WCET within the problem window.

The *workload* of a task τ_i in a problem window of length x is the amount of time in which τ_i can execute during this time interval. $W_i(x)$ denotes the maximum workload of τ_i. The *interference* from a higher priority task τ_i upon the target job within the problem window is the cumulative time in which the target job cannot execute because of the execution of τ_i.

For multicore global schedulability analysis, it is extremely hard to exactly compute the worst case interference that the target job experiences. A simplistic upper bound on the interference from a higher priority task τ_i is the maximum workload of τ_i within the problem window. However, in Bertogna et al (2005) the bound is improved by observing that the execution of τ_i that is performed in parallel with the target job should not be considered as part of the interference.

In a discrete-time system, if the target job can complete its WCET within the problem window $[a, b)$ of length x, then the interference from each higher priority task τ_i can be upper bounded as:

$$I_{i,k}(x) = \min\{W_i(x), x - C_k + 1\}.$$

This constraint is the main approach that is used in the global multicore analysis to bound the interfering workload. The unit that is added in the constraint is justified by the discrete time assumption. In continuous time, the term $(x - C_k + 1)$ can be replaced by $(x - C_k + \epsilon)$ with $\epsilon > 0$ and arbitrarily small and $I_{i,k}(x)$ can be bounded as:

$$I_{i,k}(x) = \min\{W_i(x), x - C_k + \epsilon\}. \tag{1}$$

Given the formulation of the interference in (1), multicore analysis requires that the interference is used in a condition that allows to check completion of the target job before the deadline and also that a suitable bound for the higher priority workload in (1) is defined.

As for the feasibility condition, in Baruah (2007), Γ_k is used to denote a collection of intervals, not necessarily contiguous, of cumulative length $(x - C_k)$ over $[a, b)$, during which all m cores are occupied by higher priority tasks. For each i, $1 \leq i < k$, let $I(\tau_i)$ denote the time in which a higher priority task τ_i executes in Γ_k. In order for the target job not to execute its WCET within $[a, b)$, it is necessary that the total amount of execution from higher priority tasks in Γ_k satisfies

$$\sum_{i<k} I(\tau_i) \geq m \cdot (x - C_k).$$

A sufficient condition for the feasibility of the target job in $[a, b)$ is thus

$$\sum_{i<k} I_{i,k}(x) < m \cdot (x - C_k). \tag{2}$$

This inequality is used as a *schedulability condition*. As a shortcut, we will use the notation

$$\Omega_k(x) = \sum_{i<k} I_{i,k}(x). \tag{3}$$

We assume the availability of a function $\mathcal{F}_k(\cdot)$ that returns the minimum value of x that satisfies the schedulability condition in (2) or $\mathcal{F}_k(\cdot) = \infty$ if τ_k is not schedulable:

$$\begin{gathered} \mathcal{F}_k(\cdot) = \min\{x\} \quad \text{s.t.} \\ x \geq C_k \text{ and } \Omega_k(x) < m \cdot (x - C_k). \end{gathered} \tag{4}$$

By definition, the value of $\mathcal{F}_k(\cdot)$ is also an upper bound for the WCRT of τ_k and the task is schedulable if

$$\mathcal{F}_k(\cdot) \leq D_k. \tag{5}$$

However, in many works including Guan et al (2009), instead of inequality (2), an *optimistic* schedulability condition is used for judging the schedulability of a task, as in:

$$\left\lfloor \frac{\Omega_k(x)}{m} \right\rfloor \leq x - C_k$$

that is only valid for a discrete-time assumption.

Most efforts in recent research on multicore schedulability analysis have been dedicated to the problem of finding a suitable bound for the workload $W_i(x)$.

The results in this paper are independent from the specific method that is used to upper bound $W_i(x)$. However, it is worth recalling that the problem of estimating the workload is complicated by the absence of a time critical instant (Baker (2003)). Hence, interfering tasks can be divided in two sets. Tasks that have a carry-in job that is in execution at the time when the target task arrives (or carry-in tasks, CI, see Figure 1) and tasks that do not have a carry-in job and in the worst case arrive together with the target task (or non-carry-in tasks, NC). The contribution to the workload of a task can be larger if the task is of type carry-in, that is, $W_i^{CI}(x) \geq W_i^{NC}(x)$.

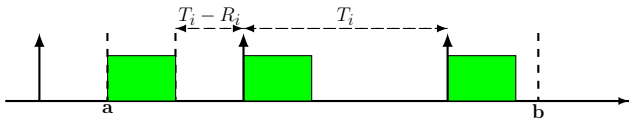


Figure 1: The worst-case arrival pattern for $W_i(x)$ with carry-in

4 Evaluating the Pessimism of Current Multicore Global Schedulability Analysis

We provide a theorem to show how global multicore FP analysis is still affected by significant pessimism. A simple task partitioning policy, denoted as $\mathcal{P}$, is used in the demonstration.

Partitioning algorithm $\mathcal{P}$: Assume a relative task priority order is defined in $\mathcal{T}$. Tasks are selected in order starting from the highest priority until the last (lowest priority) task. Each task is tentatively allocated to one of the cores, and its schedulability is checked using the single-core analysis for FP. If it is schedulable, then it is allocated to the core and its affinity is fixed. If all tasks can be allocated the algorithm succeeds, otherwise it fails.

Clearly, $\mathcal{P}$ is not optimal and quite simplistic.

Theorem 1 *Given a system with m cores, if a taskset $\mathcal{T}$ is found to be schedulable by G-FP scheduling according to (5), then the partitioning policy $\mathcal{P}$ defines a static allocation that is always schedulable.*

Proof 1 *The demonstration is by contradiction. Let us assume that the taskset $\mathcal{T}$ is schedulable by G-FP according to (5), but the partitioning defined by $\mathcal{P}$ for the tasks in $\mathcal{T}$ is not schedulable.*

The latter assumption means that the partitioning algorithm $\mathcal{P}$ fails to find a suitable core for one task. Let τ_k be the first task that fails to be allocated by $\mathcal{P}$. This means that on all the cores, the single-core analysis fails for τ_k. In single-core systems, the worst-case interference is caused when all tasks (including τ_k) are simultaneously released.

Consider a generic core p and use $\mathcal{T}_{p,k}$ to denote the set of tasks with priority higher than τ_k that have been allocated to p. Since τ_k is not schedulable, the interference it experiences is larger than $x - C_k$ for all $C_k \leq x \leq D_k$.

$$\forall p \; \forall C_k \leq x \leq D_k \quad \sum_{\tau_i \in \mathcal{T}_{p,k}} I^*_{p,i,k}(x) > x - C_k.$$

The term $I^*_{p,i,k}(x)$ denotes the single-core interference from the higher priority task τ_i within the time interval of length x starting from task τ_k's arrival time, and the index of the core p to which τ_i is allocated is added to $I^*_{p,i,k}$ for convenience. Because of the critical instant theorem, τ_i and τ_k are released simultaneously, and τ_i is an NC task.

We use $I^*_{i,k}(x)$ for the single core interference as opposed to $I_{i,k}(x)$ for the multicore case because for single-core systems it is $I^*_{i,k}(x) = W^*_i(x)$, where $W^*_i(x)$ is the single core workload and is computed according to the time critical instant.

The difference between $I^*_{i,k}(x)$ and $I_{i,k}(x)$ is that $I_{i,k}(x) = \min\{W_i(x), x - C_k + \epsilon\}$ and $W_i(x)$ can be larger than $W^*_i(x)$ when τ_i is regarded as the CI task in the multicore analysis. For the tasks that $\mathcal{P}$ allocates on a generic core p there are two possibilities.

- *1)* $W_i(x) \leq x - C_k + \epsilon$ *holds for all the tasks with higher priority than* τ_k*. In this case, for all the tasks that have been allocated to core* p*, it is*

$$\sum_{\tau_i \in \mathcal{T}_{p,k}} I_{p,i,k}(x) > x - C_k$$

in which $I_{p,i,k}(x)$ *is the interference that task* τ_i *would contribute in the global multicore case, since it is* $I^*_{i,k}(x) = W^*_i(x) \leq W_i(x) = I_{i,k}(x)$*, and according to* (1) *. We put some explanations on the relation between* $W^*_i(x)$ *and* $W_i(x)$*. There are two possibilities for the value of* $W_i(x)$*:* $W_i(x) = W^{CI}_i(x)$ *when* τ_i *has carry-in and* $W_i(x) = W^{NC}_i(x)$ *if* τ_i *is a NC task that does not bring carry-in into the problem window. As we have discussed in the end of last section, there is always* $W^{CI}_i(x) \geq W^{NC}_i(x)$*. On the other hand, in order to maximize* $W^{NC}_i(x)$*,* τ_i *shall be activated at the beginning of the problem window and all its jobs are required to arrive as soon as possible. This matches the critical instant and worst-case scenario in the single-core case and impiles that* $W^{NC}_i(x) = W^*_i(x)$*. As a result, there is* $W_i(x) \geq W^*_i(x)$*.*

- *2)* $W_i(x) > x - C_k + \epsilon$ *for at least one of the higher priority tasks allocated to* p *by* $\mathcal{P}$*. In this case, for the multicore interferences it would still be*

$$\sum_{\tau_i \in \mathcal{T}_{p,k}} I_{p,i,k}(x) > x - C_k$$

when the summation is extended to all the tasks that the policy $\mathcal{P}$ *has allocated to the core* p *up to task* τ_k*, since at least one task contributes with a term greater than* $x - C_k + \epsilon$ *to the summation on the left side.*

When you consider that none of the cores would be schedulable since $\mathcal{P}$ *terminates because it cannot find a feasible core, it is*

$$\forall p \sum_{\tau_i \in \mathcal{T}_{p,k}} I_{p,i,k}(x) > x - C_k$$

Consider now the global schedulability analysis. For any time $C_k \leq x \leq D_k$*, it is*

$$\sum_{i<k} I_{i,k}(x) = \sum_{1 \leq p \leq m} \sum_{\tau_i \in \mathcal{T}_p} I_{p,i,k}(x) > m \cdot (x - C_k).$$

hence,

$$\Omega(x) > m \cdot (x - C_k).$$

Thus, the schedulability condition in (5) *is not satisfied with any* $x \in [C_k, D_k]$ *and* τ_k*'s schedulability cannot be guaranteed according to the global FP analysis, which contradicts the original assumption.*

■

In the end, we prove that a taskset is globally schedulable only if it is schedulable by partitioned scheduling, too.

As a matter of fact, the multicore global scheduling and partitioned scheduling are incomparable: a taskset that is not schedulable under the global scheduling may or may not be schedulable by the partitioned scheduling and vice versa. Unfortunately, Theorem 1 reveals the fact that state-of-the-art analysis fails to reflect the possible advantage of multicore G-FP scheduling. The only reason for this is the excessive pessimism of the current analysis techniques.

When estimating the interference, the available schedulability analysis methods lack a fine-grained knowledge on the possible parallel execution of tasks among multiple cores. In the following, we are going to explain how to explore the parallel execution of different tasks for multicore global schedulability analysis to solve the problem highlighted in Theorem 1.

5 Improving Multicore G-FP Analysis

In case of a single-core system, when the execution of a task is preempted by a higher priority task, the task cannot resume execution until the interfering task finishes. This is not true in multicore global scheduling, where the higher priority task may only interfere with a portion of the preempted task's unfinished execution. After some time, one core may become idle and the preempted task resumes execution in parallel with the preempting task. We may use this consideration to analyze the execution of the target job in a more controllable way.

In this section, we investigate the relation between the execution of different fragments that compose the WCET of the target task. We demonstrate that the consideration of segments helps reduce the pessimism when upper bounding the interference upon the execution of the target job.

However, note that it is not the intention of this paper to develop a brand new schedulability test to replace existing ones, and the main purpose of this section is to provide people some hints on where the pessimism shown in Theorem 1 comes from.

5.1 The 2-part execution scenario

Given the target task $\tau_k = (C_k, D_k, T_k)$, we explicitly divide the execution of its jobs into two parts with worst case execution times $C_{k,1}$ and $C_{k,2}$ such that $C_{k,1}, C_{k,2} \in \mathbb{N}$ and $C_{k,1} + C_{k,2} = C_k$.

Within the problem window $[a, b)$, we define a time instant $z < b$ with $z - a \in \mathbb{N}^+$ such that τ_k completes an amount of execution over $[a, z)$ of *at least* $C_{k,1}$.

This results in the *2-part execution scenario* depicted in Figure 2. We denote $x_1 = z - a$ and $x_2 = b - z$. For convenience, we use the notion x-window to denote the time interval $[a, b)$; similarly, the x_1-window and x_2-window are defined. Given a value for $C_{k,1}$, a simple approach to compute the corresponding

x_1 is $x_1 = \lceil \mathcal{F}_{k'}(\cdot) \rceil$ (simply obtained by replacing τ_k with the virtual task $\tau_{k'} = (C_{k,1}, D_k, T_k)$.)

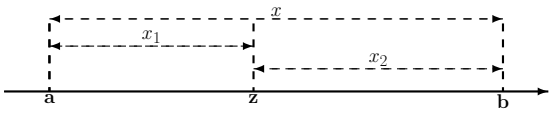


Figure 2: The 2-part execution scenario

The benefit of the 2-part execution scenario is that partial information on the target job execution (about the first part) is available, and this information can be used to improve the bound on the overall interference suffered by the target job.

Subject to the fact that τ_k finishes (at least) its first $C_{k,1}$ amount of execution time within the x_1-window, we can refine the calculation of the interference from a higher priority task τ_i upon the target job over the x-window.

Lemma 1 *Assume the 2-part execution scenario as in Figure 2, and τ_k executes for exactly $\gamma_1 \in [C_{k,1}, C_k]$[1] in the x_1-window, then the interference from a higher priority task τ_i on the target job can be bounded as:*

$$I'_{i,k}(x) = I_{i,k_1}(x_1) + \min\{I_{i,k}(x) - I_{i,k_1}(x_1), I_{i,k_2}(x_2)\} \tag{6}$$

where

$$I_{i,k_1}(x_1) = \min\{W_i(x_1), x_1 - \gamma_1\}$$

and

$$I_{i,k_2}(x_2) = \min\{W_i(x_2), x_2 - (C_k - \gamma_1)\}.$$

Proof 2 *We follow the reasoning in Section 3. Γ_k is a set of time intervals over $[a, b)$ with cumulative length $(x - C_k)$ such that inside Γ_k all cores are busy executing of higher priority tasks. The formulation of $I'_{i,k}(x)$ is required to upper bound the execution that τ_i can conduct within Γ_k.*

In the 2-part execution scenario, the set of time intervals Γ_k can be divided into two parts, the first part $\Gamma_{k,1}$ inside the x_1-window, and the second part $\Gamma_{k,2}$ inside the x_2-window. τ_k completes for γ_1 execution units inside the x_1-window. Hence, $\Gamma_{k,1}$ has an overall length of $(x_1 - \gamma_1)$ and $\Gamma_{k,2}$ is of length $(x_2 - C_k + \gamma_1)$.

Within the x_1-window, the maximum workload of a higher priority task τ_i is $W_i(x_1)$. Since τ_k executes for γ_1 units, then τ_i executes in $\Gamma_{k,1}$ for at most $I_{i,k_1}(x_1) = \min\{W_i(x_1), x_1 - \gamma_1\}$.

To compute the maximum execution of τ_i inside the $\Gamma_{k,2}$, we consider two cases.

- *If the execution of τ_i in $\Gamma_{k,1}$ is maximized, then the execution of τ_i inside $\Gamma_{k,2}$ is bounded by $I_{i,k}(x) - I_{i,k_1}(x_1)$, where $I_{i,k}(x)$ is the conventional*

[1] For simplicity, $x_1 \geq \gamma_1$ and $x_2 \geq C_k - \gamma_1$.

interference estimation as discussed in Section 3 and is re-used here. On the other hand, by the workload τ_i can generate in the x_2-window, its execution inside $\Gamma_{k,2}$ cannot exceed $\min\{W_i(x_2), x_2 - C_k + \gamma_1\}$. Thus, the maximum execution of τ_i within Γ_k is as in (6).

- *If the execution of τ_i in $\Gamma_{k,1}$ is not maximized, the value of $I_{i,k_1}(x_1)$ decreases, and $I_{i,k}(x) - I_{i,k_1}(x_1)$ increases by the same amount; because $I_{i,k_2}(x_2)$ does not change, the overall value of the interference cannot be larger than the formulation in* (6).

In the end, the formulation of the interference upper bound is as in Equation (6).

■

It is easy to see that the new interference formulation in (6) never exceeds the original one in (1).

The 2-part execution scenario allows to analyze the possible parallel execution between the target task and the interfering task in a more fine-grained approach.

The formulation in (6) considers only the case in which τ_k completes a specific portion of its execution inside the x_1-window. To verify the schedulability of τ_k, we need to consider all the possible executions of τ_k within the x_1-window.

Given a pair $C_{k,1}$ and $C_{k,2}$, the function $\mathcal{F}_k(C_{k,1})$ returns the minimum x value such that for all the possible executions of τ_k with γ_1 units ($\gamma_1 \in [C_{k,1}, C_k]$) in the x_1-window, the schedulability condition in (2), after replacing the interference term $I_{i,k}$ with the new formulation $I'_{i,k}$, is satisfied .

$$\mathcal{F}_k(C_{k,1}) = \min\{x\} \quad \text{s.t.}$$
$$x \geq C_k \text{ and } \forall \gamma_1 \in [C_{k,1}, C_k] \text{ (2) is satisfied.}$$

By definition, $\mathcal{F}_k(C_{k,1})$ is an upper bound on the response time of the target job.

5.2 Integer assumption and early termination condition

If we assume that all task parameters (including $C_{k,1}$ and $C_{k,2}$) are integers, and if we restrict to an integer upper bound on the WCRT of τ_k, the values of x_1, x_2 and x can also be assumed to be integers. Still, the scheduling decisions (e.g., preemption) are made in continuous time domain. Finally, all the available methods for bounding the workload $W_i(x)$ of a higher priority task consist of adding an integer multiple of its WCET (an integer) with a possible additional execution of a carry in instance (again, an integer) and a fractional execution right before the end of the window of length x and assuming execution at the highest rate (arrivals multiple of integer values). Hence, being the sum of integer values, also the bound on the maximum workload $W_i(x)$ is an integer value. As a result, we only need to consider the possible integer values of γ_1 for computing $\mathcal{F}_k(C_{k,1})$.

To understand why, let us assume that γ_1 increases from the integer value $\mathbf{n}$ to $\mathbf{n}+\delta$ with $\delta \in \mathbb{R}$ and $\delta \in (0,1]$, the value of $I'_{i,k}(x)$ in (6) increases to $I''_{i,k}(x)$.

In addition, the workload term $W_i(\cdot)$ and the interference term $I_{i,k}(x)$ are not affected, and the only possibilities are that $I_{i,k_1}(x_1)$ remains the same or decreases by δ, and $I_{i,k_2}(x_2)$ remains the same or increases by δ, as in the right side of Equation (6).

However, since the value of $I'_{i,k}(x)$ increases to $I''_{i,k}(x)$, the only possibility is that the value of $I_{i,k_1}(x_1)$ does not change and $I_{i,k_2}(x_2)$ increases by δ. In this case, the increase in the overall interference is $I''_{i,k}(x) - I'_{i,k}(x) = \delta$, which is maximized when $\delta = 1$.

In practice, instead of checking every integer $\gamma_1 \in [C_{k,1}, C_k]$, (suppose we go from smaller to larger values of γ_1) we can stop as soon as the following early termination condition is satisfied and returns a schedulability decision:

$$\sum_{i<k} I_{i,k_2}(x_2) < m \cdot (x_2 - C_k + \gamma_1).$$

This says that in case τ_k completes an amount of execution γ_1 within the x_1-window, and if the total interference within the x_2-window is maximized, τ_k can still complete the remaining $(C_k - \gamma_1)$ execution time within the x_2-window. According to the sustainability property (Baker and Baruah (2009b)), there is no need to further increase the γ_1 value.

5.3 An improved test

Given an arbitrary selection of $C_{k,1}$ and $C_{k,2}$, the value $\mathcal{F}_k(C_{k,1})$ is an upper bound for the WCRT of τ_k. By considering all the possible combinations of $C_{k,1}$ and $C_{k,2}$, the minimum value for $\mathcal{F}_k(C_{k,1})$ is a safe upper bound for the WCRT of τ_k.

Theorem 2 *The WCRT of the target task τ_k's is upper bounded by*

$$\min_{\forall C_{k,1} \in \mathbb{N} \wedge C_{k,1} \in [0, C_k]} \{\mathcal{F}_k(C_{k,1})\}.$$

The proposed schedulability analysis improves on the state of the art, as the new analysis in Theorem 2 strictly dominates the traditional analysis, which can be considered as a special instance of the 2-part execution scenario with $C_{k,1} = 0$ and $C_{k,2} = C_k$.

If we use $\mathbf{O}(\mathcal{F}_k(\cdot))$ to denote the complexity of the $\mathcal{F}_k(\cdot)$ function, then, computing the corresponding function $\mathcal{F}_k(C_{k,1})$ for all the possible values of γ_1 has complexity $C_k \cdot \mathbf{O}(\mathcal{F}_k(\cdot))$. The 2-part execution scenario itself does not require a new methodology to formulate the workload in the problem window, but provides a new perspective to convert the workload to interference more precisely with a limited additional cost in terms of complexity. The complexity of the final test in Theorem 2 is pseudo-polynomial. The test in Theorem 2 is regarded as a baseline test for applying the 2-part execution scenario. It considers all the C_k possibilities to divide the WCET of τ_k ($C_{k,1} = 0$ and

$C_{k,1} = C_k$ are equivalent cases). In reality, when applying the analysis on the 2-part execution scenario, heuristics for exploring first more promising values for $C_{k,1}$ and $C_{k,2}$ could be found.

6 Comparison

In this section, we show the key advantage of the new analysis based on the 2-part execution scenario. Besides, we show empirical comparison results between the new analysis and the traditional multicore G-FP schedulability analysis. In our experiments, the workload is computed following the algorithms presented in Guan et al (2009). The baseline test based on the 2-part execution scenario has been implemented in the open source software tool RTSCAN (Lipari and Sun (2013)).

6.1 An example showing the improvements in the analysis

Theorem 1 states that state-of-the-art G-FP schedulability analysis is dominated by the (suboptimal) partitioning algorithm $\mathcal{P}$ followed by single-core analysis, despite the fact that multicore global scheduling and partitioned scheduling are incomparable. Our 2-part execution analysis overcomes such a limitation for multicore global schedulability analysis since the dominance criterion of Theorem 1 does not apply to it.

As an example, let us consider a real-time system with $m = 2$ and 3 tasks: $\tau_1 = (10, 20, 20)$, $\tau_2 = (15, 30, 30)$ and $\tau_3 = (24, 50, 50)$. By the exact single core RTA in Audsley et al (1991), it is easy to check that under the partitioned FP scheduling, there is no partitioning that can guarantee all three tasks meet their deadlines. Also, according to Theorem 1, the system is also not schedulable by conventional G-FP schedulability analysis. Even if we assume discrete-time scheduling, the RTA tests in Guan et al (2009) and Sun (2015) result in an unschedulable decision.

Using the 2-part execution scenario, the system (i.e., its lowest priority task τ_3) is indeed schedulable. Below we outline the procedure to assess the schedulability of τ_3. In this simple case, most of the computations can be done manually.

The schedulability condition is met when C_3 is divided into two parts with $C_{3,1} = 9$ and $C_{3,2} = 15$. It is easy to verify that a value for x_1 such that τ_3 completes the execution of its first part $C_{3,1}$ is $x_1 = 20$,

In a time interval of length 20, $W_1(20) = 10$ and $W_2(20) = 15$. As a result, the interferences added by τ_1 and τ_2 on the execution of the first part of τ_3 are 10, and 11 ($min\{W_2(20), x_1 - C_{3,1}\}$), respectively. Given that $10 + 11 < m \cdot (20 - C_{3,1})$, τ_3 is guaranteed to complete at least $C_{k,1}$ units of execution within a window of $x_1 = 20$ time units.

Table 1 shows the interference computed according to the 2-part execution analysis under the hypothesis of $\gamma_1 = C_{3,1} = 9$. The interference in the second window of size $x_2 = 30$ of both τ_1 and τ_2 is 15 units, and the WCRT of τ_3 can be

upper bounded by $x = 50$ for $\gamma_1 = 9$. Details on how to compute the workload can be referred to Guan et al (2009).

At this point, to test the schedulability of τ_3, we need to consider all the remaining γ_1 values in the range $[C_{3,1}, C_3]$ (or until the early termination applies). When all the remaining values are considered (results are similar to those in Table 1), the schedulability of τ_3 can be verified.

τ_i	$I'_{i,3}(x)$	$I_{i,3}(x)$	$I_{i,k_1}(x_1)$	$I_{i,k_2}(x_2)$	$W_i(x)$	$W_i(x_2)$
τ_1	25	26	10	15	30	20
τ_2	26	26	11	15	30	15

Table 1: A case study for 2-part execution scenario

6.2 Performance comparison on randomly generated tasksets

In this part, we evaluate the possible advantage of the test in Theorem 2 with respect to the state-of-the-art analysis in (5) by conducting a set of experiments on randomly generated tasksets. Due to the guarantee that our test strictly dominates the traditional analysis, the analysis results are always going to improve on existing analysis methods. On the other side, as we are going to see, the improvement brought by the new analysis is very limited. Consequently, even though the new test breaks the limitation on the pessimism as shown in Theorem 1, it is not sufficient enough for practical use and more efforts are still needed for enhancing the performance of the global schedulability analysis.

We used two methods for the generation of the taskset. The first builds the set incrementally and follows the approach traditionally used in many papers on multicore analysis. The second method consists in the generation of independent tasksets.

Tasksets incrementally generated In our experiments, the system consists of $m = 16$ identical cores. The minimum inter-arrival time T_i is (uniformly) randomly selected in the range $[1, 2000]$. Each task is defined by a tuple (U_i, D_i, T_i) such that $D_i = T_i$ or D_i is randomly selected in $[C_i, T_i]$ with $C_i = U_i \cdot T_i$. Task priorities are assigned according to the deadline monotonic policy, that is, a task with a smaller deadline is given a higher priority.

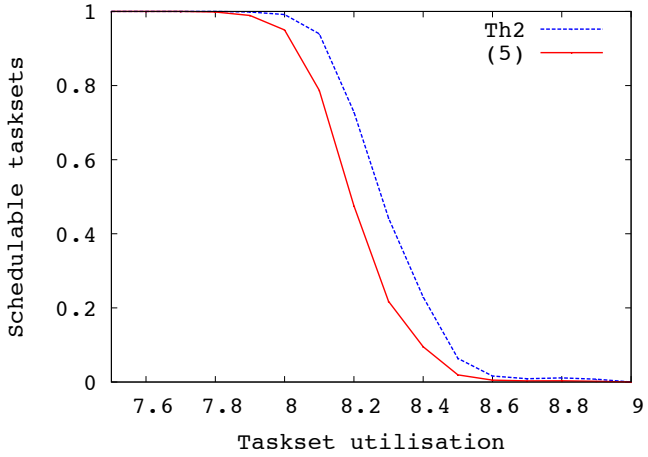
The utilization U_i of each task is extracted according to an exponential distribution with parameter $\lambda = 0.1$ and the values of U_i are clipped within the range $[U_i^{min}, U_i^{max}]$ with $U_i^{min} = 0.05$ and $U_i^{max} = 0.45$ such that when $U_i < U_i^{min}$ (resp. $U_i > U_i^{max}$), U_i is set to U_i^{min} (resp. U_i^{max}).

The taskset generation follows the approach that is commonly used (as in Bertogna and Baruah (2011), Guan et al (2009) et al.) when comparing schedulability tests for multicore global scheduling.

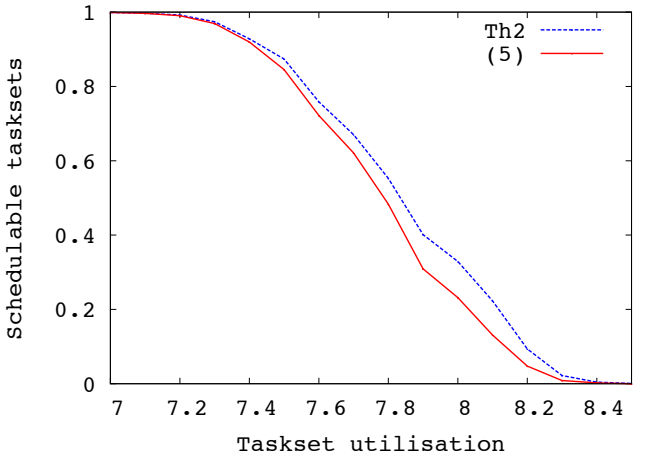
1. At first, a taskset of size $m + 1$ tasks is generated.

2. The tests in (5) and Theorem 2 are applied and the results of the schedulability analyzes recorded. If at least one test finds the taskset to be schedulable, then the taskset is extended by adding one more randomly generated task and a new analysis is performed; otherwise, a new taskset of size $m + 1$ is generated and the step 2 continues.

The generation of tasksets and the analysis comparison continues until the target number of cases is reached.



(a) $U_i^{min} = 0.05$, $U_i^{max} = 0.45$ and $D_i = T_i$



(b) $U_i^{min} = 0.05$, $U_i^{max} = 0.45$ and $D_i \leq T_i$

Figure 3: Analysis performance comparison for incrementally generated tasksets with $m = 16$

The results are shown in Figure 3, where the upper line denotes the results

from our test (in Theorem 2) and the lower line represents results from state-of-the-art analysis in (5). The results confirm a (limited) advantage for the 2-part execution analysis, as our test admits a substantial larger fraction of tasksets in a range of utilizations of width approximately 0.5. This is true for the case of implicit deadlines (Figure 3a) and, to a more limited degree, for the case of deadlines smaller than the periods (Figure 3b).

Independent tasksets Because of the way the tasksets are generated, the previous comparative analysis (as well as those that appeared in the cited papers) is biased in favor of any method that provides a possible improvement (since as soon as a task configuration provides an advantage to one method, the following set immediately provides one more opportunity to extend the advantage).

In the second set of experiments, we fix the number of tasks in each taskset to $n = 100$, and we generate the tasks in each set in order to achieve a target value for the overall system utilization (sum of utilizations of all tasks in a set). For each system utilization value (with step 0.4), we randomly generate 100 independent tasksets using the Randfixedsum algorithm in Emberson et al (2010), which ensures a fair distribution for the utilization of each task in the set. As shown in Figure 4, for independent sets and a relatively large number of tasks the advantages of our analysis are extremely small.

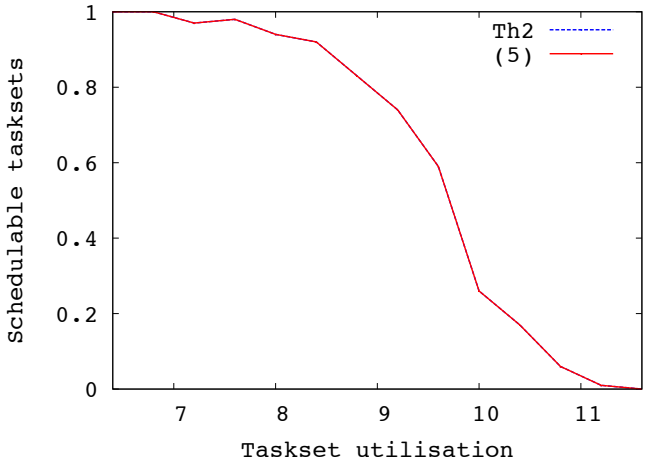


Figure 4: Another comparison with $m = 16$, $n = 100$ and $D_i = T_i$

7 Conclusion

In this work, we point out the limitation of current methodology for multicore G-FP schedulability analysis, by demonstrating its pessimism with respect to a particular strategy for partitioning tasks upon multicores. This result is the

main contribution of this paper, and we believe it helps people understand better the analysis problem in case of multicore global scheduling. Although in this paper we cannot tackle the such pessimism, we show the most likely cause of the problem and open an opportunity for improving the quality of multicore global scheduling analysis and overcoming the dominance in terms of analysis guarantees of partitioned scheduling.

References

Audsley N, Burns A, Richardson MF, Wellings AJ (1991) Hard real-time scheduling: The deadline-monotonic approach. In: in Proc. IEEE Workshop on Real-Time Operating Systems and Software, pp 133–137

Baker T (2003) Multiprocessor EDF and deadline monotonic schedulability analysis. IEEE Real-Time Systems Symposium (RTSS)

Baker T, Baruah S (2009a) An analysis of global EDF schedulability for arbitrary-deadline sporadic task systems. Real-Time Systems 43(1):3–24

Baker TP, Baruah SK (2009b) Sustainable multiprocessor scheduling of sporadic task systems. In: Real-Time Systems, 2009. ECRTS'09. 21st Euromicro Conference on, IEEE, pp 141–150

Baruah S (2007) Techniques for multiprocessor global schedulability analysis. In: Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International, pp 119–128

Baruah S, Bini E (2008) Partitioned scheduling of sporadic task systems: an ilp-based approach. In: Proceedings of the 2008 Conference on Design and Architectures for Signal and Image Processing, Citeseer

Baruah S, Bonifaci V, Marchetti-Spaccamela A, Stiller S (2009) Implementation of a speedup-optimal global EDF schedulability test. In: Real-Time Systems, 2009. ECRTS'09. 21st Euromicro Conference on, IEEE, pp 259–268

Baruah SK, Mok AK, Rosier LE (1990) Preemptively scheduling hard-real-time sporadic tasks on one processor. In: Real-Time Systems Symposium, 1990. Proceedings, 11th, IEEE, pp 182–190

Bertogna M, Baruah S (2011) Tests for global EDF schedulability analysis. Journal of Systems Architecture 57(5):487–497, special Issue on Multiprocessor Real-time Scheduling

Bertogna M, Cirinei M (2007) Response-time analysis for globally scheduled symmetric multiprocessor platforms. In: Real-Time Systems Symposium, 2007. RTSS 2007. 28th IEEE International, pp 149–160

Bertogna M, Cirinei M, Lipari G (2005) Improved schedulability analysis of EDF on multiprocessor platforms. In: Real-Time Systems, 2005.(ECRTS 2005). Proceedings. 17th Euromicro Conference on, IEEE, pp 209–218

Bonifaci V, Marchetti-Spaccamela A (2012) Feasibility analysis of sporadic real-time multiprocessor task systems. Algorithmica 63(4):763–780

Burmyakov A, Bini E, Tovar E (2015) An exact schedulability test for global FP using state space pruning. In: Proceedings of the 23rd International Conference on Real Time and Networks Systems, ACM, pp 225–234

Chen JJ (2015) Partitioned multiprocessor fixed-priority scheduling of sporadic real-time tasks. arXiv preprint arXiv:150504693

Cucu-Grosjean L, Goossens J (2011) Exact schedulability tests for real-time scheduling of periodic tasks on unrelated multiprocessor platforms. Journal of systems architecture 57(5):561–569

Davis R, Burns A (2011a) Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems. Real-Time Systems 47(1):1–40

Davis RI, Burns A (2011b) A survey of hard real-time scheduling for multiprocessor systems. ACM Computing Surveys (CSUR) 43(4):35

Emberson P, Stafford R, Davis R (2010) Techniques for the synthesis of multiprocessor tasksets. In: 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS), pp 6–11

Fisher N, Baruah SK, Baker TP (2006) The partitioned scheduling of sporadic tasks according to static-priorities. In: 18th Euromicro Conference on Real-Time Systems, ECRTS'06, 5-7 July 2006, Dresden, Germany, Proceedings, pp 118–127

Geeraerts G, Goossens J, Lindstrm M (2013) Multiprocessor schedulability of arbitrary-deadline sporadic tasks: complexity and antichain algorithm. Real-Time Systems 49(2):171–218

Guan N, Gu Z, Deng Q, Gao S, Yu G (2007) Exact schedulability analysis for static-priority global multiprocessor scheduling using model-checking. Software Technologies for Embedded and Ubiquitous Systems pp 263–272

Guan N, Stigge M, Yi W, Yu G (2009) New response time bounds for fixed priority multiprocessor scheduling. In: Proceedings of the 30th IEEE Real-Time Systems Symposium, RTSS 2009, Washington, DC, USA, 1-4 December 2009, pp 387–397

Guan N, Han M, Gu C, Deng Q, Yi W (2015) Bounding carry-in interference to improve fixed-priority global multiprocessor scheduling analysis. In: Embedded and Real-Time Computing Systems and Applications (RTCSA), 2015 IEEE 21st International Conference on, IEEE, pp 11–20

Huang WH, Chen JJ (2015) Response time bounds for sporadic arbitrary-deadline tasks under global fixed-priority scheduling on multiprocessors. In: Proceedings of the 23rd International Conference on Real Time and Networks Systems, ACM, pp 215–224

Lee J (2014) Time-reversibility of schedulability tests. In: Proceedings of the IEEE 35th IEEE Real-Time Systems Symposium, RTSS 2014, Rome, Italy, December 2-5, 2014, pp 294–303

Lee J, Shin I (2013) Limited carry-in technique for real-time multi-core scheduling. Journal of Systems Architecture 59(7):372–375

Lee J, Shin KG, Shin I, Easwaran A (2015) Composition of schedulability analyses for real-time multiprocessor systems. IEEE Trans Computers 64(4):941–954

Lipari G, Sun Y (2013) Real-Time Schedulability Analyzer (RTSCAN). Web page: https://github.com/glipari/rtscan

Liu C, Anderson JH (2013) Suspension-aware analysis for hard real-time multiprocessor scheduling. In: Real-Time Systems (ECRTS), 2013 25th Euromicro Conference on, IEEE, pp 271–281

Liu CL, Layland JW (1973) Scheduling algorithms for multiprogramming in a hard-real-time environment. Journal of the ACM (JACM) 20(1):46–61

Pathan RM, Jonsson J (2014) Interference-aware fixed-priority schedulability analysis on multiprocessors. Real-Time Systems 50(4):411–455

Spuri M (1996) Analysis of deadline scheduled real-time systems. Tech. rep.

Sun Y (2015) Real-time schedulability analysis with formal techniques. PhD thesis, Scuola Superiore S. Anna

Sun Y, Lipari G (2014) A weak simulation relation for real-time schedulability analysis of global fixed priority scheduling using linear hybrid automata. In: Proceedings of the 22nd International Conference on Real-Time Networks and Systems, ACM

Sun Y, Lipari G (2015a) A pre-order relation for exact schedulability test of sporadic tasks on multiprocessor global fixed-priority scheduling. Real-Time Systems pp 1–33

Sun Y, Lipari G (2015b) Response time analysis with limited carry-in for global earliest deadline first scheduling. In: 2015 IEEE Real-Time Systems Symposium, RTSS 2015, San Antonio, Texas, USA, December 1-4, 2015, pp 130–140

Sun Y, Lipari G, Guan N, Yi W (2014) Improving the response time analysis of global fixed-priority multiprocessor scheduling. In: Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on, IEEE, pp 1–9