Harmonicity Aware Task Partitioning for Fixed Priority Scheduling of Probabilistic Real-Time Tasks on Multi-Core Platforms

Tianyi Wang, Florida International University
Soamar Homsi, Florida International University
Linwei Niu, West Virginia State University
Shaolei Ren, University of California at Riverside
Ou Bai, Florida International University
Gang Quan, Florida International University
Meikang Qiu, Pace University

The uncertainty due to performance variations of IC chips and resource sharing on multi-core platforms have significantly degraded the predictability of real-time systems. Traditional deterministic approaches based on the worst-case assumptions become extremely pessimistic and thus unpractical. In this paper, we address the problem of scheduling a set of fixed-priority periodic real-time tasks on multi-core platforms in a probabilistic manner. Specifically, we consider task execution time as a probabilistic distribution and study how to schedule these tasks on multi-core platforms with guaranteed Quality of Service (QoS) requirements in terms of deadline missing probabilities. Moreover, it is a well known fact that the relationship among task periods, if exploited appropriately, can significantly improve the processor utilization. To this end, we present a novel approach to partition real-time tasks that can take both task execution time distributions and their period relationships into consideration. From our extensive experiment results, our proposed methods can greatly improve the schedulability of real-time tasks when compared with existing approaches.

Categories and Subject Descriptors: C.3.J.7 [**Real time**]: Real-time and embedded systems

General Terms: Design, Algorithms, Performance

Additional Key Words and Phrases: probabilistic; multi-core; task partitions; harmonic; real-time systems

1. INTRODUCTION

With the fast pace of technology scaling, billions of transistors are integrated on a single chip. As a transistor's feature size continues to shrink, to the level close to the wavelength of light used to print them, it becomes very difficult to precisely control the manufacturing process [Nassif et al. 2007]. Therefore, the performance of IC chips becomes less and less deterministic. Moreover, multi-core technology is becoming mainstream which leads to increased sharing on multi-core platforms which makes program executions less predictable. Such randomness can significantly degrade the predictability of computing systems, which is critical for real-time systems.

The traditional real-time system analysis adopts a deterministic approach, i.e. based on deterministic real-time parameters such as the worst-case execution times (WCET), and provides a deterministic guarantee [Liu and Layland 1973; Lehoczky et al. 1989; Lehoczky 1990] such that all jobs from every single task can meet their deadlines. As computing performance becomes less and less pre-

This work is supported by the National Science Foundation, under grant CNS-1423137, CNS-1457506, CNS-1565474 and ECCS-1610471.
Author's addresses: T. Wang, S. Homsi, O. Bai, and G. Quan, Department of Electrical and Computer Engineering, Florida International University, FL USA; L. Niu, Department of Mathematics and Computer Science, West Virginia State University, WV USA; S.Ren, Department of Electrical and Computer Engineering, University of California at Riverside, CA, USA; M. Qiu, Department of Computer Science, Pace University, NY, USA.

DOI:http://dx.doi.org/10.1145/0000000.0000000

dictable, such a deterministic design can lead to extremely pessimistic design. In addition, the hard deadline guarantee may not always be necessary for many real-time systems that allow a portion of the jobs miss their deadlines. For example, even for the aerospace industry, a probability of failure no more than 10^{-9} per hour is considered to be feasible by the certification authorities [ARINC 2008].

The probabilistic approach (e.g. [Tia et al. 1995]), on the other hand, is a more effective approach to tackle the randomness of real-time systems and to guarantee the design constraints from a statistical perspective. It takes probabilistic characteristics, such as probabilistic execution times, into real-time analysis and system design to meet real-time design constraints without over-provisioning [Edgar and Burns 2001].

In this paper, we are interested in the problem of how to schedule a set of fixed-priority real-time tasks with probabilistic execution times on multiple homogeneous processing cores and satisfy the given deadline miss probabilities. We focus on fixed-priority scheduling schemes since fixed-priority scheduling is one of the most popular scheduling schemes in real-time system design. It has simpler implementation and better practicability than other dynamic priority-based schedulings [Bini and Buttazzo 2005].

Given the NP-hard nature of task partitioning problem [Garey and Johnson 1990], one intuitive approach is to transform the problem into a simple bin-packing problem [Johnson et al. 1974], and employ the feasibility test methods such as those developed in [Kim et al. 2005; Maxim and Cucu-Grosjean 2013] to ensure the deadline miss probability guarantee. Note that, it is a well known fact that the period relationship among tasks, if exploited appropriately, can greatly improve the processor utilization [Fan and Quan 2012; Wang et al. 2015]. The challenge however is how to determine if a task is more "harmonic" than another one to a reference task if their periods are not strictly integer multiples, and their execution times are probabilistic instead of deterministic. To this end, we develop four novel metrics, with one improving upon another, to quantify the degree of harmonicity between two tasks. Based on these metrics, we then develop an algorithm that takes both the probabilistic execution times and task period relationship into consideration to guide the partition process for periodic tasks with random execution times on multi-core platforms. We have conducted extensive simulations to validate our approach. The experimental results show that the proposed approach can significantly improve the schedulability of real-time tasks when compared with the deterministic approach or the traditional bin-packing approaches.

The rest of the paper is organized as follows. In Section 2 we discuss related work. In Section 3 we introduce our system models and formally define our problem. In Section 4 we present the harmonic-aware metrics we developed. In Section 5 we introduce our task partition algorithm in details. Section 6 presents the experimental results and finally we conclude in Section 7.

2. RELATED WORK

In this section, we discuss the related work from two aspects: first, the studies that tackle the harmonic relationship for periodic tasks; and second, the studies that deal with probabilistic analysis of real-time scheduling.

The harmonic relationship for periodic tasks have been extensively studied for Rate Monotonic Scheduling (RMS) on uniprocessor [Bini et al. 2001; Lauzac et al. 2003; Lu et al. 2006]. It is proved that if all the tasks in a task set are harmonic (i.e., any two tasks' periods pairwise divide each other), the utilization bound can be as high as 1. Han et al. proposed a polynomial-time method to determine the feasibility of a task set by verifying the feasibility of the corresponding harmonic task set transformed from the original task set [Han and Tyan 1997]. They proved that any task set that can pass the feasibility test by *Liu&Layland's* bound can also be validated by their proposed test. Another approach by Kuo et al. [Kuo and Mok 1991] combines harmonic tasks into one task to reduce the effective number of tasks and then *Liu&Layland's* bound can be applied. More recently, Bonifaci et al. [Bonifaci et al. 2013] studied the feasibility of tasks with explicit deadlines on a uniprocessor. They proved that when all the tasks have harmonic periods, an exact polynomial-time algorithm for computing the response time of tasks can be found for both fixed priority scheduling

and dynamic priority scheduling. Nasri et al. [Nasri et al. 2014] presented a method to determine a set of harmonic periods among the possible values for tasks to simplify the worst-case timing analysis and to improve the system utilization. The advantages of exploiting task harmonic relationship have drawn many attentions due to its potential high resource utilization and low complexity on feasibility checking.

As computing applications and systems grow in scope and scale, there have been increasing interests on probabilistic approaches for real-time system analysis and design. For example, Tia et al. [Tia et al. 1995] presented a probabilistic performance guarantee approach for semi-periodic tasks on a single processor, by transforming semi-periodic tasks into a periodic task followed by a sporadic task. Atlas et al. [Atlas and Bestavros 1998] introduced a statistical rate monotonic scheduling for periodic tasks with statistical QoS requirements. Maxim et al. [Maxim et al. 2011] proposed three priority assignment algorithms for probabilistic real-time systems. They further improved the previous work by proposing a framework of re-sampling mechanism that can simplify the calculation of response time distributions in order to ease timing analysis for real-time systems in [Maxim et al. 2012]. Yue et al. [Lu et al. 2012] presented a statistical response time analysis by analysing samples in timing traces taken from real systems. In [Kim et al. 2005], the authors proposed a stochastic analysis framework which computed the response time distribution and deadline miss probability for each individual task. The framework can be applied to both fixed-priority and dynamic-priority systems on a single-core platform. The authors in [Maxim and Cucu-Grosjean 2013] extended their work to allow both task's execution time and period to be random variables and computed analytically the response time distribution of the tasks on uniprocessor under a task-level fixed-priority preemptive scheduling policy. In [Axer and Ernst 2013], the authors proposed a new convolution-based stochastic analysis method for single processor fixed-priority non-preemptive scheduling policy to bound the response time under fault situations.

For multi-core case, Li et al. [Li and Tang 2013] proposed a task scheduling algorithm for heterogeneous computing systems considering deadline and energy consumption budget constraints. They studied the problem of scheduling a bag-of-tasks application, with independent stochastic tasks of normal distribution as task execution times. In [Li et al. 2013], the authors presented a model of scheduling stochastic parallel applications on heterogeneous cluster systems. They proposed a scheduling algorithm for parallel applications based on stochastic bottom levels and stochastic dynamic levels. None of these approaches takes task period relationship into consideration.

We believe that by taking advantage of the harmonic relationship between periodic tasks can greatly improve the processor usage and system feasibility. Several previous researches (e.g. [Guan et al. 2012; Kandhalu et al. 2012]) have already exploited the harmonic relationship between periodic tasks by applying so called *R-Bound* [Lauzac et al. 1998], i.e., a utilization bound that takes the possible harmonic relationship into consideration. More recently, Fan et al. [Fan and Quan 2014] proposed a semi-partitioned approach for fixed-priority tasks on multi-core platforms with harmonic relationship exploration. They formally proved that any task set with a system utilization bounded by *Liu&Layland's* bound can be successfully scheduled. However, all these approaches assume deterministic parameters in their task models, such as worst-case execution times (which could lead to overly pessimistic results). Therefore, these approaches cannot readily apply to solve the problem when the task execution times are probabilistic. In our approach, we are interested in partitioning tasks with statistical execution times by exploiting harmonic relationship among them. We develop four metrics, with each one improving upon another, to quantitatively measure the harmonic relationship between different tasks with statistical execution times. Then based on the proposed four metrics, we allocate tasks that are close to be harmonic to one processor to improve system resource usage. To our best knowledge, this is the first paper that tackles task mapping with probabilistic execution times on multi-core systems by taking advantage of harmonic relationship [Wang et al. 2016].

3. PRELIMINARY

In this section we first introduce our system models including real-time task models and processor models. We then formulate the problem formally.

3.1. System models and problem formulation

We consider a real-time system consisting of N independent periodic tasks, denoted as $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$, to be scheduled on a homogeneous multi-core platform, denoted as $\mathcal{P} = \{p_1, p_2, ..., p_K\}$, according to the RMS policy. Each task $\tau_i \in \Gamma$, is characterized by a tuple $(\mathcal{C}_i, T_i)$, where

$$\mathcal{C}_i = \begin{pmatrix} c_1 = c_{min} & ...c_k & \cdots & c_n = c_{max} \\ Pr(c_{min}) & ...Pr(c_k) & \cdots & Pr(c_{max}) \end{pmatrix} \tag{1}$$

representing *the worst-case execution time distribution* [Edgar and Burns 2001; Lu et al. 2012] of τ_i, i.e., the probability that the worst execution time of $\mathcal{C}_i = c_k$ is $Pr(c_k)$. For all possible values of $\mathcal{C}_i$, we have $c_k \in [c_{min}, c_{max}]$, where c_{min} and c_{max} are the minimum and maximum values for $\mathcal{C}_i$, and $\sum_{k=1}^{n} Pr(c_k) = 1$. T_i is the period of task τ_i and we have $T_i \leq T_j$ if $i < j$. We also assume that the deadline of a task equals its period, i.e., $D_i = T_i$.

Since a task's execution time is not unique, the response time for each job may be different. Therefore a job may meet or miss its deadline. We formally define the concept of *deadline miss probability* as follows.

Definition 1. *The deadline miss probability (DMP) of job $\tau_{i,j}$ (denoted as $DMP_{i,j}$) is the probability that job $\tau_{i,j}$ misses its deadline and can be formulated as following:*

$$DMP_{i,j} = Pr(\mathcal{R}_{i,j} > D_{i,j}) \tag{2}$$

where $\mathcal{R}_{i,j}$ is the response time distribution of job $\tau_{i,j}$ and $D_{i,j}$ is the deadline of job $\tau_{i,j}$. Accordingly, the deadline miss probability of task τ_i (denoted as DMP_i) is defined as the probability that the task misses its deadline and formulated as

$$DMP_i = Pr(\mathcal{R}_i > D_i) \tag{3}$$

Finally, the deadline miss probability for a task set Γ (denoted as DMP_Γ) is defined as

$$DMP_\Gamma = max\{DMP_i\}, \tau_i \in \Gamma. \tag{4}$$

Our research problem can be formulated as follows:

Problem 1. *Given*

— a task set consisting of N tasks, $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$,
— a multicore platform with K homogeneous processing cores, $\mathcal{P} = \{p_1, p_2, ..., p_K\}$,
— and the deadline miss probability constraint, i.e. $DMP_\Gamma \leq C_d$, with C_d a constant

partition the task set Γ on the multi-core platform and schedule the tasks on each core using RMS scheme such that the deadline miss probability constraint for the task set is satisfied and the number of cores is minimized.

3.2. Motivations

One simple approach to solve this problem is to transform it into the traditional bin-packing problem. Note that, with the knowledge of tasks assigned to a processing core, Yue et al. [Lu et al. 2012] proposed a method to calculate the probabilistic response time distribution for a real-time task under a preemptive uniprocessor fixed-priority scheduling policy, which can be applied to determine if the DMP constraint can be satisfied. Therefore, traditional bin-packing approaches such as First Fit, Next Fit, Best Fit can be readily applied to assign tasks to different cores.

It is a well known fact that, for RMS, the processor utilization can reach as high as one if tasks are harmonic, i.e. task periods are integer multiples of one another. For tasks that are not entirely

harmonic, Fan et al. [Fan et al. 2015] showed that, if the period relationships among tasks can be appropriately exploited, the processor utilization can be significantly improved. Specifically, they introduced three interesting concepts, *sub harmonic task set*, *the primary harmonic task set*, and *harmonic index*, which are defined as follows:

Definition 2. *[Fan et al. 2015] Given a task set* $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$*, let* $\hat{\Gamma} = \{\hat{\tau_1}, \hat{\tau_2}, \ldots, \hat{\tau_N}\}$*, where* $\hat{\tau}_i = (C_i, \hat{T}_i)$*,* $\hat{T}_i \leq T_i$*, and* $\hat{T}_i | \hat{T}_j$ *if* $i < j$ *(*$a|b$ *means "a divides b" or "b is an integer multiple of a"). Then* $\hat{\Gamma}$ *is a sub harmonic task set of* Γ*.*

Definition 3. *[Fan et al. 2015] Let* Γ' *be a sub harmonic task set of* Γ*. Then* Γ' *is called a* primary harmonic task set *of* Γ *if there exists no other sub harmonic task set* Γ'' *such that* $T_i' \leq T_i''$ *for all* $1 \leq i \leq N$*.*

Definition 4. *[Fan et al. 2015] Given a task set* Γ*, let* $\mathcal{G}(\Gamma)$ *represent all the primary harmonic task sets of* Γ*. Then the* harmonic index *of* Γ*, denoted as* $\mathcal{H}(\Gamma)$*, is defined as*

$$\mathcal{H}(\Gamma) = \min_{\Gamma' \in \mathcal{G}(\Gamma)} \Delta(U') \tag{5}$$

where

$$\Delta(U') = \begin{cases} U(\Gamma') - U(\Gamma) & \text{if } U(\Gamma') \leq 1, \\ +\infty & \text{otherwise.} \end{cases} \tag{6}$$

$U(\Gamma)$ and $U(\Gamma')$ represent the utilizations of task set Γ and Γ', respectively. We can employ *Sr* or *DCT* algorithm [Han et al. 1996; Han and Tyan 1997] to find all sub harmonic task sets with a complexity as low as $O(N \cdot log(N))$.

We believe that, by exploiting the period relationships among tasks, we can greatly improve the processor utilization. The challenge is how to quantify the degree of harmonicity among different tasks with probabilistic execution times. For tasks with deterministic execution times, according to Definition 4, given a reference task, a task with its original period closer to the transformed period in the primary harmonic task set has a higher degree of harmonicity. However, the degree of harmonicity of a task to its reference task may depend not only on its period but also on its execution time distribution as well. Consider a task set with three tasks $\tau_a = \{\binom{2}{0.3}\ \binom{3}{0.7}, 6\}$, $\tau_b = \{\binom{4}{0.5}\ \binom{6}{0.5}, 12\}$, and $\tau_c = \{\binom{3}{0.5}\ \binom{7}{0.5}, 12\}$. Note that both τ_b and τ_c have the same period and same mean execution time. If we allocate τ_a and τ_b together to a core, we have $DMP_{\tau_a,\tau_b} = 0$. If we allocate τ_a and τ_c to a core, we have $DMP_{\tau_a,\tau_c} = 24.5\%$. Therefore, the degree of harmonicity of a task depends not only on its period, but also on its execution time distribution as well.

In what follows, we first introduce four metrics that we have developed, with each improving upon the previous one, to quantify the degree of harmonicity between two tasks. We then propose an algorithm that takes both the probabilistic execution times and task period relationships into consideration to guide the partition process for periodic tasks with random execution times on multi-core platforms.

4. HARMONIC INDEX FOR TASKS WITH PROBABILISTIC EXECUTION TIMES

In this section, we formally introduce the metrics which take the harmonic relationship into consideration to guide our allocation of tasks with random execution times. Since not all tasks in a task set are strictly harmonic, it is desirable that we quantify the *harmonicity* of tasks and allocate tasks with higher degree of harmonicity to the same processor to achieve higher utilization as well as higher schedulability. In what follows, we develop four metrics to measure the harmonic relationship among tasks.

4.1. Mean-based harmonic index

Our goal is to quantify the degree of harmonicity between two tasks. Before we define the harmonic index for this purpose, we first introduce the following concept.

Definition 5. *Given a task set* $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ *and one of its primary harmonic task set* $\Gamma' = \{\tau'_1, \tau'_2, \ldots, \tau'_N\}$*, let* $\tau_r \in \Gamma$*,* $\tau'_r \in \Gamma'$ *and* $\tau_r = \tau'_r$*. Then* τ_r *is called the* reference task *of the primary harmonic task set* Γ'*, and* Γ' *is called the primary harmonic task set based on* τ_r *and is denoted as* $\Gamma'(\tau_r)$*.*

According to Definition 5, the primary harmonic task set based on τ_r, i.e. $\Gamma'(\tau_r)$, is simply the primary harmonic task set with τ_r unchanged.

When task execution times are probabilistic, one intuitive approach is to employ the execution time mean and thus transform the probabilistic execution time distribution into a deterministic value. The harmonic index can therefore be defined in a similar way as that for tasks with deterministic execution times.

Definition 6. *Given a task* $\tau_i = \{C_i, T_i\} \in \Gamma$ *and its reference task* $\tau_r \in \Gamma$*, let* $\tau'_i = \{C_i, T'_i\}$ *be the corresponding task of* τ_i *in* $\Gamma'(\tau_r)$*. Then the* mean-based harmonic index *of task* τ_i *w.r.t. the reference task* τ_r*, denoted as* $\mathcal{H}_m(\tau_i, \tau_r)$*, is defined as*

$$\mathcal{H}_m(\tau_i, \tau_r) = |\bar{U}(\tau_i) - \bar{U}(\tau'_i)|, \tag{7}$$

where

$$\bar{c}_i = \sum_{\forall (c_k, Pr(c_k)) \in C_i} c_k \cdot Pr(c_k), \tag{8}$$

$$\bar{U}(\tau_i) = \frac{\bar{c}_i}{T_i}. \tag{9}$$

Note that τ_r in equation (7) indicates $\bar{U}(\tau'_i)$ is calculated under its corresponding task set $\Gamma'(\tau_r)$.

			Transformed based on τ_1 $(T_1 \vert T_i)$		Transformed based on τ_2 $(T_2 \vert T_i)$	
τ_i	(C_i, Pr_i)	T_i	$\hat{T}_i$	$\mathcal{H}_m(\tau_i, \tau_1)$	$\hat{T}_i$	$\mathcal{H}_m(\tau_i, \tau_2)$
1	$(2, 0.3)$ $(3, 0.7)$	6	6	0	5	0.09
2	$(4, 0.5)$ $(5, 0.5)$	10	6	0.3	10	0
3	$(4, 0.5)$ $(6, 0.5)$	12	12	0	10	0.083
4	$(8, 0.7)$ $(10, 0.3)$	20	18	0.032	20	0

Table I: Sub-harmonic task set transformations of a 4-task set.

Let us consider the example shown in Table I. A task set contains four independent periodic tasks, each with a probabilistic execution time distribution and a deterministic period. We transform the original task set into two primary harmonic task sets, based on τ_1 and τ_2, respectively (for more details, please check [Han and Tyan 1997]). For the first primary harmonic task set which is transformed based on task τ_1, we take τ_2 as an example to show how we derive its mean based harmonic index $\mathcal{H}_m(\tau_2, \tau_1)$. According to equation (7), $\bar{U}(\tau_2) = 0.45$ and $\bar{U}(\tau'_2) = 0.75$. Therefore, $\mathcal{H}_m(\tau_2, \tau_1) = |\bar{U}(\tau_2) - \bar{U}(\tau'_2)| = 0.3$. Then if we sort the tasks based on $\mathcal{H}_m(\tau_i, \tau_1)$, we have $\mathcal{H}_m(\tau_1, \tau_1) = 0$, $\mathcal{H}_m(\tau_3, \tau_1) = 0$, $\mathcal{H}_m(\tau_4, \tau_1) = 0.032$ and $\mathcal{H}_m(\tau_2, \tau_1) = 0.3$. If we combine task τ_1 with τ_3, task τ_1 with τ_4 and task τ_1 with τ_2 into three different subsets and allocate each subset to a processor, the deadline miss probability of each individual subset are $DMP_{\tau_1,\tau_3} = 0\%$, $DMP_{\tau_1,\tau_4} = 10.29\%$ and $DMP_{\tau_1,\tau_2} = 24.5\%$. This shows that smaller $\mathcal{H}_m$ does imply better harmonic relationship between two tasks.

From Definition 6, we can see that the mean-based harmonic index ($\mathcal{H}_m$), with a computational complexity of $O(|\mathcal{C}_i|)$ with $|\mathcal{C}_i|$ representing the number of different possible worst-case execution times of task τ_i, quantifies the harmonic relationship of a task to its reference task by measuring the difference between its expected utilization and that in the primary harmonic task set. While the mean value is a good representative value for a probabilistic distribution, it cannot capture the entire characteristics of a probabilistic distribution. Recall the example shown in Sub-section 3.2, task τ_b and τ_c do not only have the same period but also the same mean. According to $\mathcal{H}_m$, the two tasks have the same harmonic index. However, the scheduling results are different ($DMP_{\tau_a,\tau_b} \neq DMP_{\tau_a,\tau_c}$). Therefore, more effective harmonic index needs to be developed.

4.2. Variance-based harmonic index

As we have discussed earlier, it is possible that two tasks with the same mean may have different harmonic relationships compared to a third task. The reason is that the harmonic relationship refers to the "distance" between the two tasks (the smaller the "distance", the better harmonic relationship the two tasks have). However, the mean-based harmonic index calculates the "distance" solely based on the expected worst-case execution time which cannot accurately capture the harmonic relationship between two tasks. We need a harmonic metric that can account for the deviations of execution time distributions between two tasks instead of the mean execution times only, when evaluating the harmonic relationship, i.e., the "distance". It is therefore reasonable to take the variance into consideration when designing the harmonic metric. To this end, we develop a variance-based harmonic index as follows.

Definition 7. *Given $\tau_i \in \Gamma$ and its reference task $\tau_r \in \Gamma$, let $\tau_i' = \{\mathcal{C}_i, T_i'\}$ be the corresponding task of τ_i in $\Gamma'(\tau_r)$. Then the* variance-based harmonic index *of task τ_i w.r.t. the reference task τ_r, denoted as $\mathcal{H}_v(\tau_i,\tau_r)$, is defined as*

$$\mathcal{H}_v(\tau_i,\tau_r) = \mathcal{H}_m(\tau_i,\tau_r) + Var(\tau_i,\tau_r), \tag{10}$$

where

$$Var(\tau_i,\tau_r) = \frac{\sqrt{\sum_{\forall c_k \in \mathcal{C}_i}(c_k - \bar{c}_i)^2 \cdot Pr_i}}{T_i'} \tag{11}$$

$\mathcal{H}_v$, with the same computational complexity as that of $\mathcal{H}_m$, improves upon $\mathcal{H}_m$ by taking both the mean value of execution times as well as their variances into consideration. For the example shown in Section 3.2, we have $\mathcal{H}_v(\tau_a,\tau_b) < \mathcal{H}_v(\tau_a,\tau_c)$ (since $\mathcal{H}_m(\tau_a,\tau_b) = \mathcal{H}_m(\tau_a,\tau_c)$ and $Var(\tau_a,\tau_b) < Var(\tau_a,\tau_c)$) indicating that task τ_b is more harmonic than τ_c to reference task τ_a. This conforms to the results from the schedulability analysis. The variance-based harmonic index can capture more accurate harmonic relationship than mean-based harmonic index since it considers both mean and variance of execution time distributions between two tasks, and therefore, it can be a more accurate representative in terms of the harmonic relationship. However, there are still problems with the proposed harmonic metric. First, it essentially implies that both execution time mean values and variances are equally important in evaluating the degree of harmonicity. Second, again, using only mean value and its variance cannot capture accurately the characteristics of execution time distributions. Many execution time distributions may have the same mean value and variance but totally different probabilistic characteristics. In what follows, we quantify the harmonic relationship between two tasks based on their execution time distribution formulations.

4.3. Cumulative distribution function based harmonic index

We believe that we can achieve a better correlation of harmonic index and task schedulability if we can capture execution time distributions more accurately and incorporate them into the harmonic index. To this end, we propose another metric developed on the cumulative distribution function of task execution times. Before we present our new harmonic index, we first introduce the following concepts and notations.

Definition 8. *Given $\tau_i \in \Gamma$, the cumulative distribution function of the task's utilization, denoted as $CDF_{\tau_i}(u)$, can be formulated as*

$$CDF_{\tau_i}(u) = \sum Pr(\frac{C_i}{T_i} \leq u) \tag{12}$$

Essentially, the cumulative distribution function is the utilization CDF of task τ_i. Note that CDFs for $\tau_i \in \Gamma$ and its corresponding task $\tau_i' \in \Gamma$ are different. To measure the "distance", we can use the ℓ^2-*norm* operation.

Definition 9. *Given a vector* $\mathbf{x} = [x_1, x_2, ..., x_p]$*, its ℓ^2-norm, denoted as $||\mathbf{x}||$, is defined as*

$$||\mathbf{x}|| = \sqrt{\frac{\sum_{k=1}^{p} |x_k|^2}{p}} \tag{13}$$

Now we are ready to define the new harmonic index.

Definition 10. *Given $\tau_i \in \Gamma$ and its reference task $\tau_r \in \Gamma$, let $\tau_i' = \{C_i, T_i'\}$ be the corresponding task of τ_i in the primary harmonic task set of Γ, i.e. $\tau_i' \in \Gamma'(\tau_r)$. Then the* cumulative distribution function based harmonic index *of task τ_i to τ_r, denoted as $\mathcal{H}_C(\tau_i, \tau_r)$, is defined as*

$$\mathcal{H}_{cdf}(\tau_i, \tau_r) = ||CDF_{\tau_i}(\mathbf{x}) - CDF_{\tau_i'}(\mathbf{x})|| \tag{14}$$

where $\mathbf{x} = [x_1, x_2, ..., x_p]$ represents a sequence of utilization values and

$$CDF_{\tau_i}(\mathbf{x}) = [CDF_{\tau_i}(x_1), CDF_{\tau_i}(x_2), ..., CDF_{\tau_i}(x_p)], \tag{15}$$

$$CDF_{\tau_i'}(\mathbf{x}) = [CDF_{\tau_i'}(x_1), CDF_{\tau_i'}(x_2), ..., CDF_{\tau_i'}(x_p)]. \tag{16}$$

Note that for a given task τ_i and its reference task τ_r, the corresponding primary harmonic task τ_i' may have a different set of possible utilizations than that in τ_i. For example, consider a task with a four possible WCETs, i.e. $\tau_i = \{\binom{3}{0.3}\ \binom{4}{0.3}\ \binom{6}{0.2}\ \binom{8}{0.2}, 12\}$. Accordingly we have four possible utilizations $\binom{x_1}{0.25}\ \binom{x_2}{0.33}\ \binom{x_3}{0.50}\ \binom{x_4}{0.67}$ and their corresponding CDFs as $CDF_{\tau_i}(x_1) = 0.3$, $CDF_{\tau_i}(x_2) = 0.6$, $CDF_{\tau_i}(x_3) = 0.8$ and $CDF_{\tau_i}(x_4) = 1.0$. Assume the period for the reference be 10, and we have τ_i' = $\{\binom{3}{0.3}\ \binom{4}{0.3}\ \binom{6}{0.2}\ \binom{8}{0.2}, 10\}$. Thus the possible utilizations of τ_i' become $\binom{x_1'}{0.3}\ \binom{x_2'}{0.4}\ \binom{x_3'}{0.6}\ \binom{x_4'}{0.8}$ and $CDF_{\tau_i'}(x_1') = 0.3$, $CDF_{\tau_i'}(x_2') = 0.6$, $CDF_{\tau_i'}(x_3') = 0.8$ and $CDF_{\tau_i'}(x_4') = 1.0$. Since $x_j \neq x_j', j = 1,2,3,4$, we cannot apply equation (14) directly to calculate $\mathcal{H}_{cdf}(\tau_i, \tau_r)$. To resolve this problem, we combine both (x_1, x_2, x_3, x_4) and (x_1', x_2', x_3', x_4') to form a new utilization vector, i.e. $\mathbf{x} = \{x_1, x_1', x_2, x_2', x_3, x_3', x_4, x_4'\}$ for this example, and reconstructed CDF_{τ_i} and $CDF_{\tau_i'}$ as shown in the following table. We can then

	x_1	x_1'	x_2	x_2'	x_3	x_3'	x_4	x_4'
$Utilizations(\mathbf{x})$	0.25	0.30	0.33	0.40	0.50	0.60	0.67	0.80
$CDF_{\tau_i}(\mathbf{x})$	0.30	0.30	0.6	0.60	0.80	0.80	1.00	1.00
$CDF_{\tau_i'}(\mathbf{x})$	0.0	0.30	0.30	0.60	0.60	0.80	0.80	1.00

apply equation (14) to calculate $\mathcal{H}_{cdf}(\tau_i, \tau_r)$.

The rationale behind the definition of *cumulative distribution function based harmonic index* is that we intend to determine the degree of harmonicity by measuring how much the task utilization distribution has changed after changing its period to be an integer multiple of that of the reference task. The larger the difference, the less harmonic the two tasks are. It is not difficult to see that the computational complexity for calculating $\mathcal{H}_C(\tau_i, \tau_r)$ is still $O(|C_i|)$, as $p \leq |C_i|$ in equation (14).

4.4. The utilization sum based harmonic index

Note that $\mathcal{H}_{cdf}$ determines if τ_i is harmonic to τ_r only by the "distance" of utilization distributions of task τ_i in Γ and $\Gamma'(\tau_r)$. While the utilization of τ_i can affect the task schedulability, the combined utilization distribution of τ_i and τ_r can be a better indicator to the schedulability for task sets containing both τ_i and τ_r. Therefore, to design a harmonic index that can be a more effective schedulability indicator, it is reasonable to use the sum of utilization of both τ_i and τ_r rather than that of τ_i alone.

For ease of our presentation, we first introduce the following notation.

Definition 11. *Given $\tau_i, \tau_j \in \Gamma$, the cumulative distribution function of the total utilization of τ_i and τ_j, denoted as $CDF_{\tau_i,\tau_j}(u)$, can be formulated as*

$$CDF_{\tau_i,\tau_j}(u) = \sum Pr(\frac{C_i}{T_i} + \frac{C_j}{T_j} \leq u). \tag{17}$$

CDF_{τ_i,τ_j} can be easily calculated using convolution once (C_i, T_i) and (C_j, T_j) are given.

Definition 12. *Given a task τ_i and a* reference task τ_r*, let $\tau_i' = \{C_i, T_i'\}$ be the corresponding task of τ_i in the primary harmonic task set of Γ, i.e. $\tau_i' \in \Gamma'(\tau_r)$. Then the* utilization sum based harmonic index *of τ_i based on reference task τ_r, denoted as $\mathcal{H}_{sum}(\tau_i, \tau_r)$, is defined in the following equation,*

$$\mathcal{H}_{sum}(\tau_i, \tau_r) = ||CDF_{\tau_i,\tau_r}(x) - CDF_{\hat{\tau}_i,\tau_r}(x)|| \tag{18}$$

The index $\mathcal{H}_{sum}$ evaluates the task harmonic relationship based on the variation of combined utilization distribution, which is more closely related to the task schedulability and thus can potentially achieve better results. In the meantime, however, the computational complexity also increases. Note that $CDF_{\tau_i,\tau_j}(u)$ may have as many as $|C_i| \times |C_j|$ different values. The complexity of calculating $\mathcal{H}_{sum}$ with regard to τ_i, τ_j is thus $O(|C_i| \times |C_j|)$.

So far we presented four metrics for quantifying the harmonic relationship among tasks. It forms the basis for our task partitioning approach that exploits harmonic relationship for tasks with probabilistic execution times. In what follows, we present our task partitioning algorithm in details.

5. TASK PARTITIONING

In this section, we first present our task partitioning algorithm and then we discuss how to select the best candidate task set to allocate to a core.

5.1. Partitioning algorithm

With the harmonic indices defined above, we are ready to introduce our task partitioning algorithm. Essentially, our algorithm intends to identify the tasks with the highest harmonic index values, and put them into one core to improve the processor utilization. To satisfy the DMP requirement, we conduct the schedulability analysis based on the technique proposed in [Lu et al. 2012]. The detailed algorithm is illustrated in Algorithm 1.

As shown in Algorithm 1, our algorithm chooses the reference task from the first task τ_1 till the last task τ_N. For each reference task, all the rest of the tasks are ordered according to the chosen harmonic index values, and selected from high value to low to form a sub-task set until the *DMP* test is failed. After we identify all candidate subsets from each primary harmonic task set, we choose the best subset from all the candidate subsets and allocate them to a core (to be explained further in Section 5.2). Then we delete these tasks from task set Γ. We repeat this process for the rest of the tasks until all tasks are assigned.

The computational complexity of the algorithm comes from the following major components: (1) To identify all primary harmonic task sets with complexity of $O(N^2)$, where N is the number of tasks; (2) To sort the tasks in a primary task set based on a designated harmonic index with complexity of $O(NlogN \times C_h)$, where C_h is the computational complexity for calculating the harmonic index; (3) For each primary harmonic task set, to determine how many tasks can fit in a core without violating the DMP constraint with a complexity of $O(NC_f)$, where C_f is the computational complexity for

feasibility checking of a task set. The overall complexity is therefore $O(N^3 + N^2 logN \mathcal{C}_h + N^3 \mathcal{C}_f)$. Since $\mathcal{C}_f >> N$ [Lu et al. 2012] and $\mathcal{C}_f >> \mathcal{C}_h$, the overall complexity of the Algorithm 1 is thus approximately $(N^3 \mathcal{C}_f)$.

Algorithm 1 Stochastic task partitioning algorithm.

Input:
Task set: $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$;
Task set deadline miss probability: DMP_Γ;
Output:
Task partitions: $= \{subset_1, subset_2, \ldots, subset_K\}$, K is the total number of cores.

while $\Gamma \neq \emptyset$ **do**
  $SubSet = \emptyset$; //initialize the subset to empty
  $\Gamma' = \{\Gamma'(\tau_1), \Gamma'(\tau_2), \ldots, \Gamma'(\tau_L)\}$, // identify all the primary harmonic task sets
  for $i = 1$ to N // for each primary harmonic task set **do**
    $\Gamma'(\tau_i) = \{\tau_1^{'}, \tau_2^{'}, \ldots, \tau_N^{'}\}$ // sort the tasks with increasing order of $\mathcal{H}_m$, $\mathcal{H}_v$, $\mathcal{H}_{cdf}$ or $\mathcal{H}_{sum}$
    $subset_i = \emptyset$ // initialize a subset for $\Gamma'(\tau_i)$
    for $j = 1$ to N **do**
      $subset_i = subset_i + \tau_j^{'}$
      if $DMP_{subset_i} > DMP_\Gamma$ **then**
        $subset_i = subset_i - \tau_j^{'}$;
        break;
      end if
    end for
    Identify the best $subset_i$;
    $SubSet = subset_i$;
  end for
  $\Gamma = \Gamma - SubSet$;
end while

5.2. Best subset selection

In Algorithm 1, with each task as the reference, we can identify a sub task set that is most harmonic to the reference task. All of them can satisfy the DMP requirement if they are assigned to a core. The question is then which sub task set is the *best one* and should be chosen to assign to a core. If a task utilization is a deterministic value, we can simply choose the task set with the highest total utilization. However, what if the task utilization is probabilistic rather than deterministic? We explain how we choose the best subset with an example.

Consider the example shown in Table I. Let us take task τ_1 as the reference task, then the two corresponding subsets, (τ_1, τ_3) and (τ_2, τ_4), are both schedulable with DMP = 0. Now the question is which one should we pick first to assign to a core such that the processor utilization is optimized. The intuitive idea is to pick the subset with the larger utilization. However, since the tasks have random execution times, how should we determine if a subset has a larger utilization than another?

Let $\mathcal{U}_i$ denote the utilization distribution for task τ_i. For tasks in Table I we have $\mathcal{U}_1 = \left(\begin{smallmatrix} \frac{2}{6} & \frac{3}{6} \\ 0.3 & 0.7 \end{smallmatrix}\right)$, $\mathcal{U}_2 = \left(\begin{smallmatrix} \frac{4}{10} & \frac{5}{10} \\ 0.5 & 0.5 \end{smallmatrix}\right)$, $\mathcal{U}_3 = \left(\begin{smallmatrix} \frac{4}{12} & \frac{6}{12} \\ 0.5 & 0.5 \end{smallmatrix}\right)$, and $\mathcal{U}_4 = \left(\begin{smallmatrix} \frac{8}{20} & \frac{10}{20} \\ 0.7 & 0.3 \end{smallmatrix}\right)$. The utilization distribution of a subset is the convolution of each task within the subset. So we have $\mathcal{U}_{1,3} = \mathcal{U}_1 \bigotimes \mathcal{U}_3 = \left(\begin{smallmatrix} 0.67 & 0.83 & 1.0 \\ 0.15 & 0.5 & 0.35 \end{smallmatrix}\right)$. Similarly we have $\mathcal{U}_{2,4} = \left(\begin{smallmatrix} 0.8 & 0.9 & 1.0 \\ 0.35 & 0.5 & 0.135 \end{smallmatrix}\right)$ (transformed to decimal values for better illustration). As we can see,

$\mathcal{U}_{1,3}$ and $\mathcal{U}_{2,4}$ have different possible utilization values and it is hard to determine which task set has a *larger* utilization. In what follows, we propose two methods for sub task set selection.

Method 1: Mean-based sub task set selection (MTS)

Given two subsets Γ_1 and Γ_2, let their total utilization distributions be $\mathcal{U}_{\Gamma_1}$ and $\mathcal{U}_{\Gamma_2}$. Then one intuitive approach is to rank Γ_1 and Γ_2 based on the mean values of their total utilization distributions, i.e. $\overline{\mathcal{U}_{\Gamma_1}}$, $\overline{\mathcal{U}_{\Gamma_2}}$. The higher the mean utilization for a task set is, the higher the average resource demand it requires and thus should be chosen first.

For example, since the mean utilization for subset (τ_1,τ_3) is $0.67*0.15+0.83*0.5+1*0.35=0.8655$ and that for subset (τ_2,τ_4) is $0.8*0.35+0.9*0.5+1*0.135=0.865$. We choose subset (τ_1,τ_3) over (τ_2,τ_4). This approach is simple and straightforward. However, this approach suffers the same problem as our mean-based harmonic index introduced before — the mean value cannot accurately capture the characteristic of a probabilistic distribution.

Method 2: Utilization threshold-based sub task set selection(UTTS)

A larger value of mean utilization for a task set indicates a higher workload demand in average by the task set. However, when considering the deadline miss probability, not all the workload demands are "critical" in the same way. Recall that, for fixed-priority tasks, any task set with total utilization no more than the *Liu&Layland's* bound, i.e. close to 0.7 as the task number approaching infinite, are guaranteed to be schedulable. When task set utilization is higher than the *Liu&Layland's* bound, they are not necessarily schedulable at all.

Consider two sub task sets Γ_1 and Γ_2, and let $Pr(\mathcal{U}_{\Gamma_1}>0.7)>Pr(\mathcal{U}_{\Gamma_2}>0.7)$. Since both Γ_1 and Γ_2 are guaranteed to meet the deadline miss probability requirement, it seems reasonable to choose Γ_1 first as it has a higher probability to accommodate workload demands which may not be schedulable otherwise. Based on this observation, we develop the second sub task set selection method, which is formulated in Algorithm 2.

Algorithm 2 The utilization threshold-based sub task set selection (UTTS).

```
1: Probability = 0, SubSet = subset_1; //initialization
2: Ut = 0.7;
3: for each subset_i do
4:    if Pr(U_subset_i > Ut) > Probability then
5:       Probability = Pr(U_subset_i > Ut);
6:       SubSet = subset_i;
7:    end if
8: end for
9: return SubSet;
```

As an example, we can calculate that the probability that subset (τ_1,τ_3) has utilization higher than 0.7 is $Pr(\mathcal{U}_{1,3}>0.7)=0.85$ while that for subset (τ_2,τ_4) is $Pr(\mathcal{U}_{2,4}>0.7)=1.0$. Therefore subset (τ_2,τ_4) is a better choice first because it has higher probability to have larger utilization than subset (τ_1,τ_3).

6. EXPERIMENTAL RESULTS

In this section, we used experiments to investigate the effectiveness of our proposed algorithms. We first studied how different sub task set selection strategies may affect the task partitioning performance. We then examined how effective are the four harmonic indexes presented in Section 4, when compared with other existing approaches. We collected two sets of experimental results for this purpose: (1) the required numbers of cores for given real-time task sets by different approaches; (2) the feasibility ratios of randomly generated real-time tasks on multi-core platforms with pre-defined core numbers. Finally we investigate the computational costs of our proposed approaches.

6.1. Experimental setup

In our experiments, we randomly generated real-time tasks as our test cases, using algorithm similar to UUniFast approach [Bini and Buttazzo 2005]. Specifically, we picked 2, 4, 6 and 8 different possible task's worst-case execution times for each real-time task. The possible worst-case execution times for a task was randomly picked from interval [1,100]. As shown in equation (1), our approaches are not limited to any specific probabilistic distribution for the worst-case execution time. The corresponding probability for each possible worst-case execution time of each task was also randomly generated, with overall probabilities equal to 1. We then generated periods for all the tasks randomly such that the expected utilization of each task is within range $[0.1, 0.4]$.

Six different approaches were implemented in our experiment. The first approach is the traditional bin-packing approach. Specifically, real-time tasks are ordered according to their mean utilizations, and then allocated to a core according to the first-fit strategy until the deadline miss probability exceeds the given threshold. We denote this approach as **FF**. Note that this approach does not take the task period relationship into consideration. The second approach, similar to that in [Fan et al. 2015], is the deterministic harmonicity-aware task partitioning approach based on the longest worst-case execution time and not the worse case execution time probabilistic distribution. We denote this approach as **WCET-H**. We also implemented four approaches based on four harmonic indexes we developed, i.e. *the mean-based harmonic index*, *the variance-based harmonic index*, *the cumulative distribution based harmonic index*, and *the cumulative distribution based harmonic index*, which are denoted as H_m, H_v, H_{cdf} and H_{sum}, respectively.

6.2. Subset selections

First we want to study how different sub task selection strategies may affect the task partitioning results. We implemented both strategies in Section 5.2 and compared the results by task partitioning approach H_{sum}. We denote the results using *the mean-based sub task set selection* method as MTS. For the second sub task set selection method $UTTS$, we set the utilization threshold (Ut) to one of the five values: 0.5, 0.6, 0.7, 0.8, 0.9. Specifically, we use $UTTS$-Ut to denote the results of with utilization threshold Ut. For example, $UTTS$-0.5 denotes the results when $Ut = 0.5$. Three sets of different test cases were generated, i.e. with 8 tasks, 12 tasks and 16 tasks in each task set, respectively. For each set of test cases, we randomly generated 100 task sets and set $DMP_\Gamma = 5\%$. The total number of required cores were collected and are shown in Figure 1.

From Figure 1, we can see that task partitioning results by $UTTS$-0.7, $UTTS$-0.8, and $UTTS$-0.9 outperform that by MTS in most cases. For example, in Figure 1(b) for the test cases with 8 task number and 8 possible worst-case execution times, $UTTS$-0.9 outperforms MTS by as much as 5%. Also, when considering $UTTS$ approach with different utilization thresholds, it is interesting to observe from Figures 1(a), 1(b), and 1(c) that, the the performance keeps improving as the threshold increases from 0.5 to 0.9. For example, as shown in Figure 1(b), the performance improves by almost 1 core as the threshold increases from 0.5 to 0.9. This verifies our hypothesis that selecting sub task sets with high probability of high utilizations tends to achieve better results than the ones with larger expected utilizations, as a task with high probability to reach high utilization is difficult to be schedulable together with other tasks, even though its expected utilization is relatively low.

As shown in Figure 1, the utilization threshold of 0.9 works well in $UTTS$ under different scenarios and can be easily justified. Therefore in what follows, we choose 0.9 as the utilization threshold.

6.3. Performance w.r.t. number of cores

Next, we study the performance of our proposed approaches in terms of number of cores when scheduling given task sets. We generated three types of task sets, i.e. with 8 tasks, 12 tasks and 16 tasks, respectively. For each type of task set, we also set the possible worst-case execution times for each task to be 2, 4, 6 or 8. We also studied two different deadline miss probabilities: $DMP_\Gamma = 5\%$ and $DMP_\Gamma = 10\%$. For each test case, we randomly generated 100 task sets and the results are shown in Figure 2 and Figure 3.

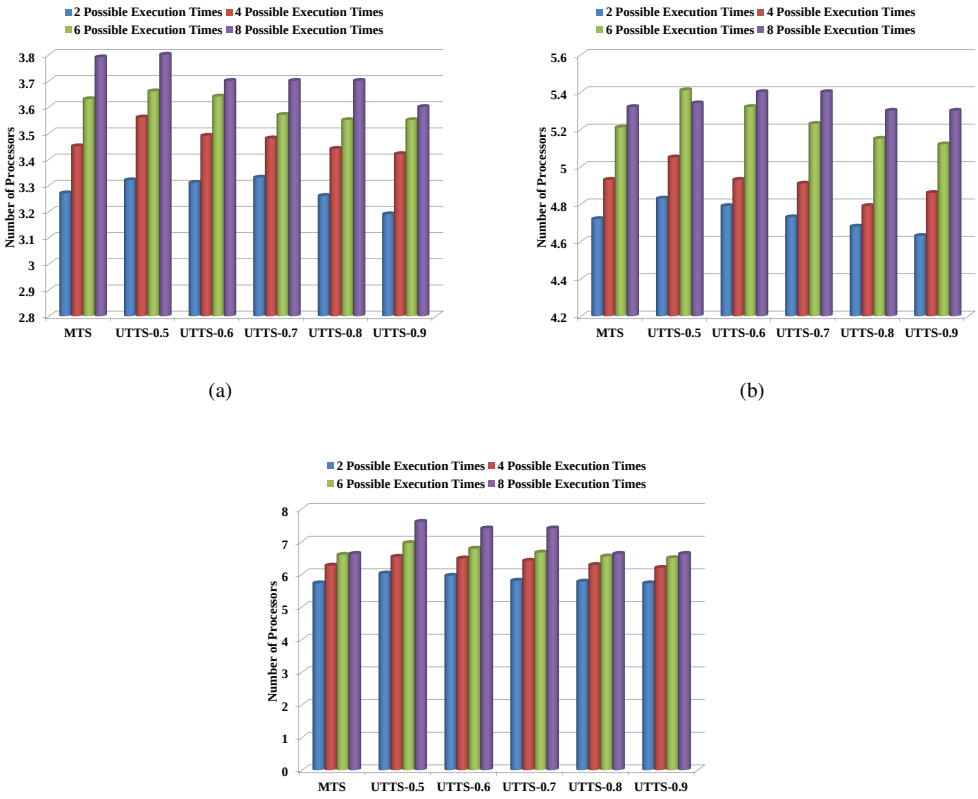

Fig. 1: Performance of subset selection methods w.r.t. number of cores for (a) 8 tasks, (b) 12 tasks and (c) 16 tasks using $DMP_{\Gamma} = 5\%$ and utilization uniformly generated in the range $[10\% - 40\%]$

From Figure 2 and Figure 3, we can see that H_m, H_v, H_{cdf} and H_{sum} outperform FF in most test cases. For example, as shown in Figure 3(c), when the number of tasks is 16, and the number of different possible worst-case execution times is 6, we can see that H_m, H_v, H_{cdf} and H_{sum} outperform FF by $0.1, 0.2, 0.3$, and 0.4 cores, respectively. This clearly shows the advantage of taking the task period relationship into consideration when partitioning tasks. In the meantime, we can also see that H_m, H_v, H_{cdf} and H_{sum} can achieve better task partitioning results than $WCET$-H, i.e. with improvement of 0.3, 0.5, 0.6 and 0.7 cores, respectively, as shown in Figure 2(c) for sets with 16 tasks and 8 worst-case execution times. $WCET$-H makes task partitioning decisions based solely on one single deterministic parameter, i.e. the longest worst-case execution times. It is thus biased and can be extremely pessimistic for task sets that exhibit significant randomness in its worst-case execution time. The other four approaches, by taking probabilistic nature of tasks into consideration, can therefore achieve better performance.

It is interesting to see that our experiments indeed show that our proposed four approaches, i.e. H_m, H_v, H_{cdf} and H_{sum}, improve one another following the same order. For example, as shown in 3(b), H_{sum} has the highest performance improvement over FF**,** $WCET$-H with 0.3 and 0.35 less core usage in average, respectively. H_m, on the other hand, has the lowest performance improvement over FF**,** $WCET$-H with 0.1 and 0.15 less core usage in average, respectively. H_m determines

how harmonic two tasks are based solely on their expected worst-case execution times. While H_v improves upon H_m by incorporating the worst-case execution time variances, the expected value and variance alone are not enough to capture the characteristics of an arbitrary probabilistic distribution accurately. H_{cdf} and H_{sum} are more elaborative and determine the harmonic relationship based on the entire probabilistic distribution of a task's worst-case execution time, and therefore become more effective to quantify how harmonic two tasks are.

In addition, our experiment results show that our proposed approaches work better with the increase of task number. As shown in Figure 3(a), H_{sum} improves upon FF by as much as 0.2 cores for test cases with 8 tasks and 8 possible worst-case execution times. When increasing the task number to 16, we can see in Figure 3(c) that H_{sum} improves upon FF by as much as 0.5 cores. Increasing task number increases the design space. Our proposed approaches can effectively optimize the task partitioning results, and therefore can achieve better results as design space increases.

It is also interesting to see that, with the increase of possible number of worst-case execution times for each task, H_{cdf} and H_{sum} improves much rapidly than H_m, and H_v. For example, in Figure 2(b), for test cases with 12 tasks and 4 different worst-case execution times, H_{sum} improves H_m by 0.15 cores; for test cases with 12 tasks and 8 different worst-case execution times, we found that H_{sum} improves H_m by 0.3 cores. When more possible worst-case execution times a task has, its statistical execution nature can be captured more accurately. This is the reason why H_{cdf} and H_{sum}, by employing the entire worst-case execution time distribution, can be more effective in determining the harmonicity between two tasks.

From Figure 2 and Figure 3, it is not surprising to see that as deadline miss probability increases, the numbers of cores needed decrease. This is due to the fact that the tighter the DMP constraint, the more resources (processing cores in this case) is needed in order to guarantee the statistical timing behaviors of real-time tasks. Our experimental results also show that our approaches perform better for systems with tighter DMP requirements. For example, for test cases with 12 tasks and 8 execution times, H_{sum} can improve upon FF by 0.3 cores when $DMP_\Gamma = 10\%$, as shown in Figure 3(c), compared with a performance improvement of 0.4 cores when $DMP_\Gamma = 5\%$, as shown in Figure 2(c). This is because that a tighter DMP implies less allocation opportunities for naive methods (FF and $WCET$-H), while our approaches can identify task allocations with better resource usage.

6.4. Performance w.r.t. schedulability

Next, we analyze the performance of different approaches in terms of schedulability. Specifically, with a given core number, we randomly generated real-time task sets and compared the number of task sets that can be successfully scheduled by different approaches. We used the same test cases as in the second experiment and set the core number to be 3, 4 and 6 for 8 tasks, 12 tasks and 16 tasks, respectively. The percentage of the schedulable task sets are shown in Figure 4 and Figure 5.

Similar conclusions can be drawn from these two figures. First, we can see that our proposed approaches can significantly improve FF and $WCET$-H up to 22% by H_{sum} for test cases with 16 tasks and 8 different worst-case execution times, as shown in Figure 5(c). Second, we can see that the feasibility ratios improvement increases following the order of H_m, H_v, H_{cdf} and H_{sum}. As shown in Figure 4(b), we can see that the schedulability improves from 37% using H_m to 48% using H_{sum}. Third, we can also see that the improvement of our proposed approaches increases with the number of tasks as well as the numbers of possible worst-case execution times for each task. For example, H_{sum} improves from 16% for test cases using 12 tasks and 8 possible worst-case execution times to 30% for test cases using 16 tasks and 8 possible worst-case execution times as shown in Figures 4(b), and 4(c), respectively. Whereas FF improves by as much as 3% for the same test cases. Finally, we can also see that our approaches work better with lower DMP. For example, the average feasibility percentage improvement of H_{sum} over FF using 8, 12 and 16 tasks with 8 possible worst-case execution times using $DMP_\Gamma = 10\%$ is 13%, as shown in Figure 5, whereas the average feasibility percentage improvement of H_{sum} over FF using same test cases with $DMP_\Gamma = 5\%$ is 12.5%. It shows that with tighter DMP, our approaches can still make better allocation choice and keep up with the improvement over traditional heuristics.

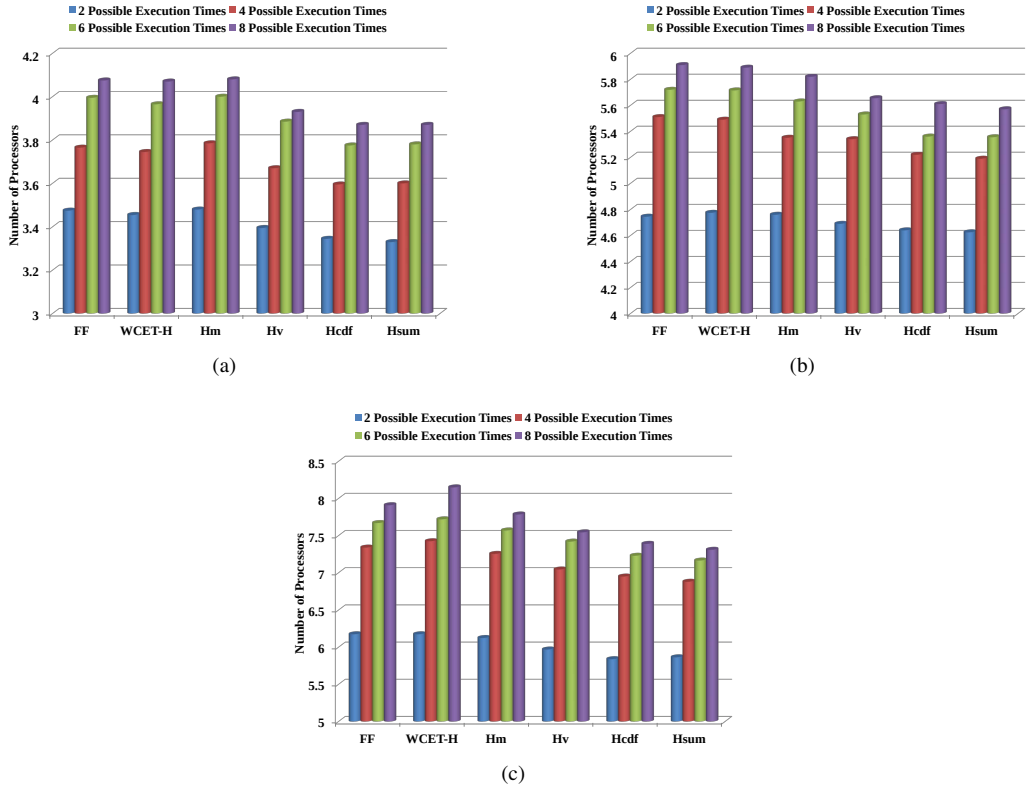

Fig. 2: Processor usage w.r.t. number of cores for (a) 8 tasks, (b) 12 tasks and (c) 16 tasks using $DMP_\Gamma = 5\%$ and utilization uniformly generated in the range $[10\% - 40\%]$

6.5. Computational Costs

Finally, we want to compare the computational costs of approaches with different harmonic indexes. Three sets of different test cases were generated, using 8 tasks, 16 tasks and 24 tasks. For each set of test cases, we randomly generated 100 task sets with 8 different worst-case execution times for each task, and set $DMP_\Gamma = 10\%$. The results are shown in Figure 6.

From Figure 6 we can see that the more tasks we have, the more time it takes for our harmonic-indexes-based approaches. For example, for 8 tasks, less than 4 seconds are needed for each approach to complete its computation. Whereas for 24 tasks, the fastest completion time is around 132 seconds (our approach with mean-based harmonic index, H_m). Moreover, H_{cdf} and H_{sum} have higher computational costs than H_m and H_v. Because H_{cdf} and H_{sum} need to calculate the difference between two distributions which takes more time compared with the mean and variance calculations. Note that, FF can finish within 3 seconds for 24 tasks. In fact, the computation costs of all four metrics are very low. Most of the computation time is spent on feasibility checking to pick the best sub task set. However, our approach is essentially an off-line scheme, and can thus tolerate relative large computational cost for its distinctive advantage to minimize resource usage while satisfying the DMP constraints.

7. CONCLUSIONS

With the increase of performance variations in modern computer systems, it is imperative to adopt a probabilistic approach rather than the traditional deterministic approach in the design and analysis

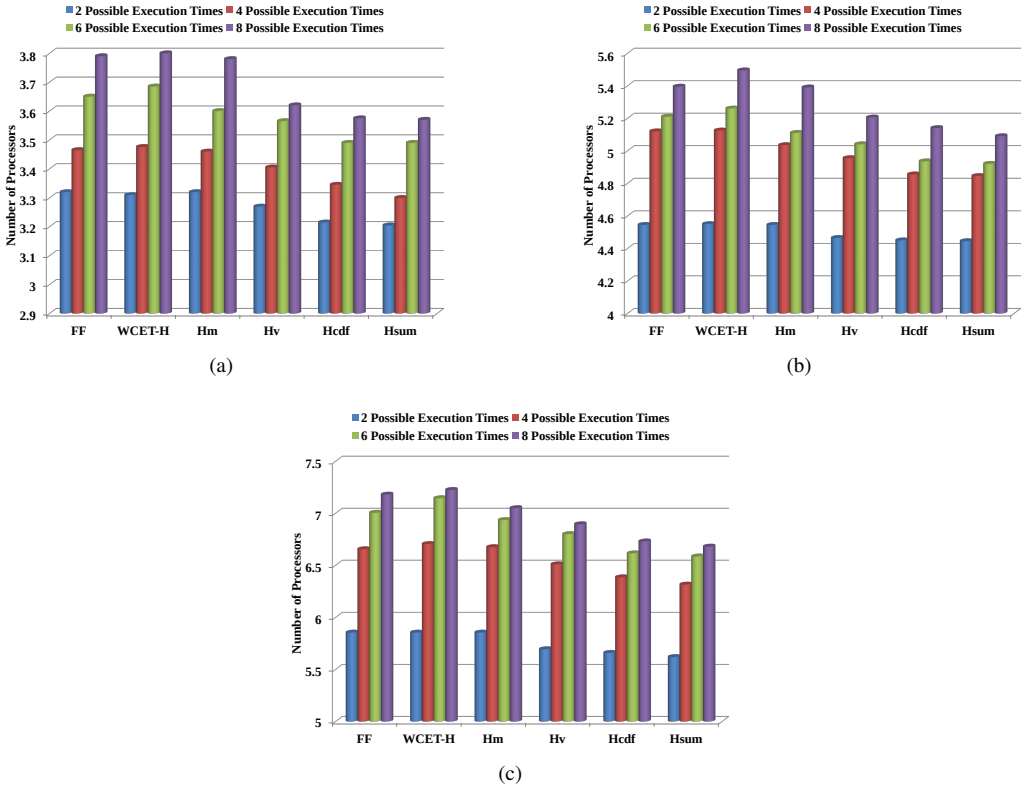

Fig. 3: Processor usage w.r.t. number of cores for (a) 8 tasks, (b) 12 tasks and (c) 16 tasks using $DMP_{\Gamma} = 10\%$ and utilization uniformly generated in the range $[10\% - 40\%]$

of real-time systems. In this paper, we developed a novel task partitioning algorithm for fixed-priority scheduling of real-time tasks with probabilistic execution times on a homogeneous multi-core platform with statistical guarantee. In our approach, we develop four novel metrics: *mean based*, *variance based*, *cumulative distribution based*, and *distribution sum based* harmonic indices to quantify the harmonicity among tasks, and based on which to better identify task set allocations and improve processor utilization. We conducted extensive simulation studies and the results show that our algorithms can significantly outperform the existing approach.

Our approach presented in this paper is focused on fixed-priority preemptive scheduling of independent real-time tasks on multi-core systems. This approach can be readily extended to several other scheduling problems such as fixed-priority non-preemptive scheduling [Marouf and Sorel 2011] and fixed-priority scheduling with resource sharing [Sha et al. 1990] as our approach can help to improve the resource utilization under those scheduling algorithms as well. Note that our approach are limited only to synchronized periodic tasks with implicit deadlines, i.e. deadlines are equal to their periods. Therefore, even though the fixed-priority scheduling of periodic tasks with data dependency can be transformed to that of fixed-priority scheduling of independent tasks [de Oliveira and da Silva Fraga 2000], our approach cannot apply since the transformed task sets are no longer synchronized task sets and/or have implicit deadlines. How to extend our approach to task sets with constrained deadline is an interesting problem and will be our future study.

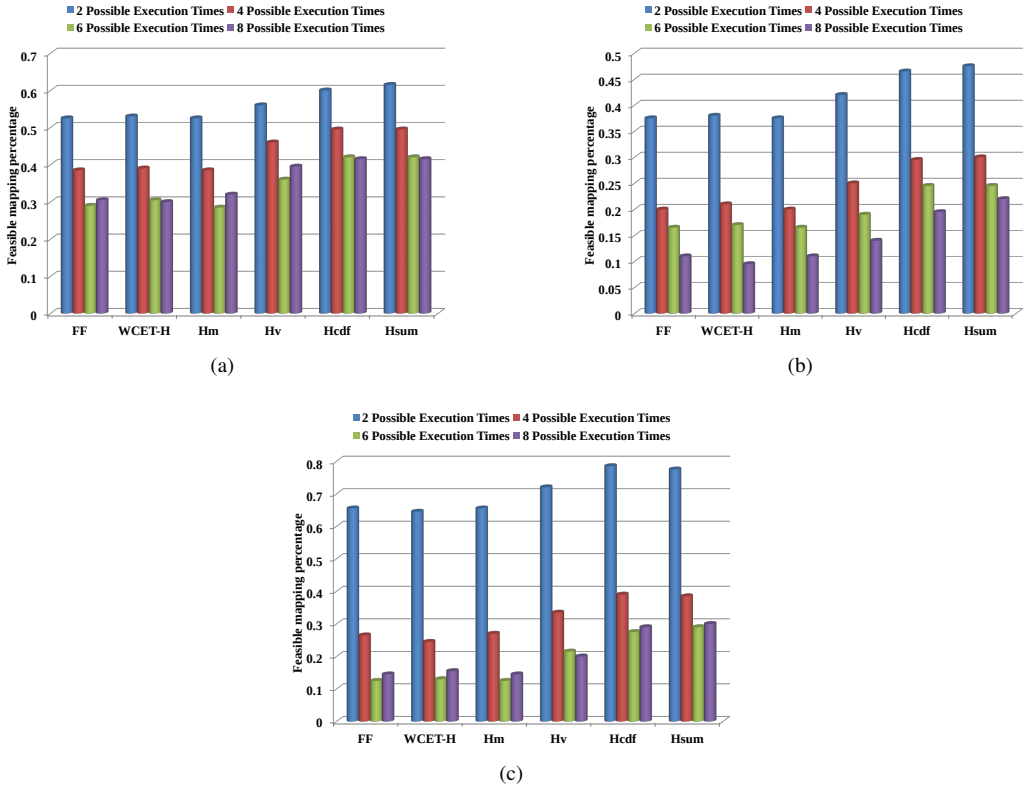

Fig. 4: Feasible mapping percentages for different approaches using (a) 8 tasks, (b) 12 tasks and (c) 16 tasks with $DMP_{\Gamma} = 5\%$ and utilization uniformly generated in the range $[10\% - 40\%]$

REFERENCES

ARINC. 2008. An avionics standard for safe, partitioned systems. In *Wind River Systems/IEEE Seminar*.

A Atlas and A Bestavros. 1998. Statistical rate monotonic scheduling. In *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*. 123–132. DOI:http://dx.doi.org/10.1109/REAL.1998.739737

P. Axer and R. Ernst. 2013. Stochastic response-time guarantee for non-preemptive, fixed-priority scheduling under errors. In *Design Automation Conference (DAC), 2013 50th ACM / EDAC / IEEE*. 1–7.

Enrico Bini, Giorgio Buttazzo, and Giuseppe Buttazzo. 2001. A Hyperbolic Bound for the Rate Monotonic Algorithm. In *Proceedings of the 13th Euromicro Conference on Real-Time Systems (ECRTS '01)*. IEEE Computer Society, Washington, DC, USA, 59–. http://dl.acm.org/citation.cfm?id=871910.871919

Enrico Bini and Giorgio C. Buttazzo. 2005. Measuring the Performance of Schedulability Tests. *Real-Time Syst.* 30, 1-2 (May 2005), 129–154. DOI:http://dx.doi.org/10.1007/s11241-005-0507-9

Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, Nicole Megow, and Andreas Wiese. 2013. Polynomial-time exact schedulability tests for harmonic real-time tasks. In *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*. IEEE, 236–245.

Romulo Silva de Oliveira and Joni da Silva Fraga. 2000. Fixed priority scheduling of tasks with arbitrary precedence constraints in distributed hard real-time systems. *Journal of Systems Architecture* 46, 11 (2000), 991 – 1004. DOI:http://dx.doi.org/10.1016/S1383-7621(00)00004-7

S. Edgar and A Burns. 2001. Statistical analysis of WCET for scheduling. In *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE*. 215–224. DOI:http://dx.doi.org/10.1109/REAL.2001.990614

Ming Fan, Qiushi Han, Shuo Liu, Shaolei Ren, Gang Quan, and Shangping Ren. 2015. Enhanced fixed-priority real-time scheduling on multi-core platforms by exploiting task period relationship. *Journal of Systems and Software* 99 (2015), 85–96.

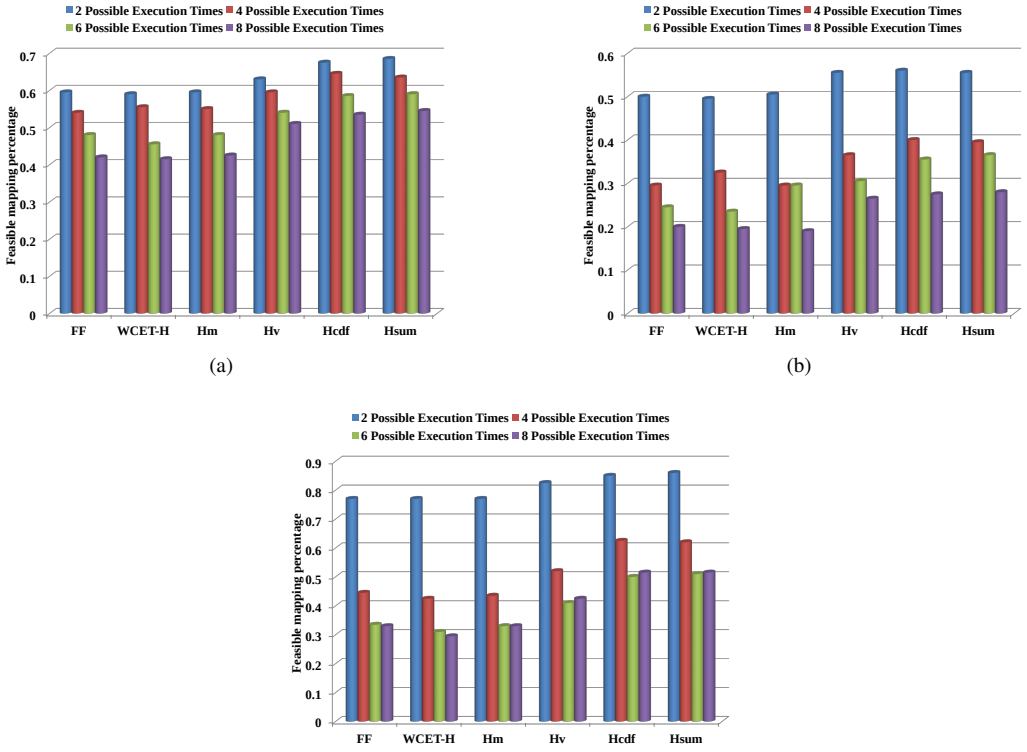

Fig. 5: Feasible mapping percentages for different approaches using (a) 8 tasks, (b) 12 tasks and (c) 16 tasks with $DMP_{\Gamma} = 10\%$ and utilization uniformly generated in the range $[10\% - 40\%]$

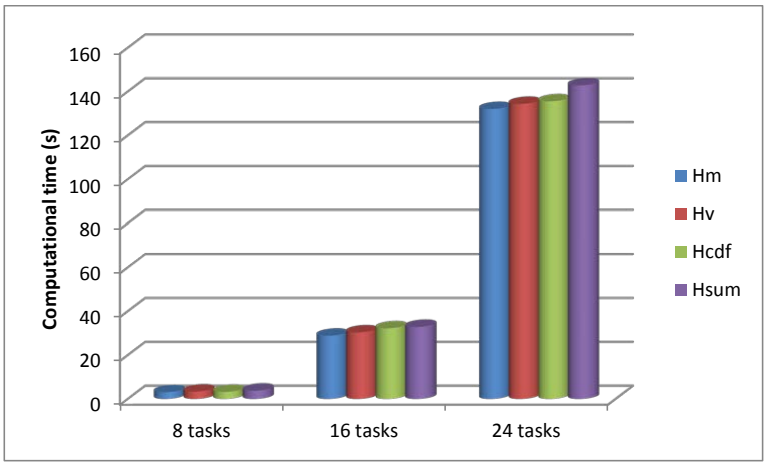

Fig. 6: Computational costs of approaches with different harmonic indexes with $DMP_{\Gamma} = 10\%$

Ming Fan and Gang Quan. 2012. Harmonic semi-partitioned scheduling for fixed-priority real-time tasks on multi-core platform. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*. 503–508. DOI:http://dx.doi.org/10.1109/DATE.2012.6176521

Ming Fan and Gang Quan. 2014. Harmonic-Aware Multi-Core Scheduling for Fixed-Priority Real-Time Systems. *Parallel and Distributed Systems, IEEE Transactions on* 25, 6 (June 2014), 1476–1488. DOI:http://dx.doi.org/10.1109/TPDS.2013.71

Michael R. Garey and David S. Johnson. 1990. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA.

Nan Guan, Martin Stigge, Wang Yi, and Ge Yu. 2012. Parametric utilization bounds for fixed-priority multiprocessor scheduling. In *Parallel & Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International*. IEEE, 261–272.

Ching-Chih Han, Kwei-Jay Lin, and Chao-Ju Hou. 1996. Distance-Constrained Scheduling and Its Applications to Real-Time Systems. *IEEE Trans. Comput.* 45, 7 (July 1996), 814–826. DOI:http://dx.doi.org/10.1109/12.508320

Ching-Chih Han and Hung-ying Tyan. 1997. A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*. IEEE, 36–45.

D. Johnson, A. Demers, J. Ullman, M. Garey, and R. Graham. 1974. Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms. *SIAM J. Comput.* 3, 4 (1974), 299–325. DOI:http://dx.doi.org/10.1137/0203025

Arvind Kandhalu, Karthik Lakshmanan, Junsung Kim, and Ragunathan Rajkumar. 2012. pCOMPATS: period-compatible task allocation and splitting on multi-core processors. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*. IEEE, 307–316.

Kanghee Kim, J.L. Diaz, L. Lo Bello, J.M. Lopez, Chang-Gun Lee, and Sang-Lyul Min. 2005. An exact stochastic analysis of priority-driven periodic real-time systems and its approximations. *Computers, IEEE Transactions on* 54, 11 (Nov 2005), 1460–1466. DOI:http://dx.doi.org/10.1109/TC.2005.174

Tei-Wei Kuo and A.K. Mok. 1991. Load adjustment in adaptive real-time systems. In *Real-Time Systems Symposium, 1991. Proceedings., Twelfth*. 160–170. DOI:http://dx.doi.org/10.1109/REAL.1991.160369

Sylvain Lauzac, Rami Melhem, and Daniel Mossé. 1998. An efficient RMS admission control and its application to multiprocessor scheduling. In *Parallel Processing Symposium, 1998. IPPS/SPDP 1998. Proceedings of the First Merged International... and Symposium on Parallel and Distributed Processing 1998*. IEEE, 511–518.

Sylvain Lauzac, Rami Melhem, and Daniel Mossé. 2003. An Improved Rate-Monotonic Admission Control and Its Applications. *IEEE Trans. Comput.* 52, 3 (March 2003), 337–350. DOI:http://dx.doi.org/10.1109/TC.2003.1183948

J.P. Lehoczky. 1990. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *Real-Time Systems Symposium, 1990. Proceedings., 11th*. 201–209. DOI:http://dx.doi.org/10.1109/REAL.1990.128748

J. Lehoczky, Lui Sha, and Y. Ding. 1989. The rate monotonic scheduling algorithm: exact characterization and average case behavior. In *Real Time Systems Symposium, 1989., Proceedings.* 166–171. DOI:http://dx.doi.org/10.1109/REAL.1989.63567

Kenli Li and Xiaoyong Tang. 2013. Energy-Efficient Stochastic Task Scheduling on Heterogeneous Computing Systems. (2013).

Kenli Li, Xiaoyong Tang, and Bharadwaj Veeravalli. 2013. Scheduling Precedence Constrained Stochastic Tasks on Heterogeneous Cluster Systems. (2013).

C. L. Liu and James W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *J. ACM* 20, 1 (Jan. 1973), 46–61. DOI:http://dx.doi.org/10.1145/321738.321743

Wan-Chen Lu, Hsin-Wen Wei, and Kwei-Jay Lin. 2006. Rate Monotonic Schedulability Conditions Using Relative Period Ratios. In *Embedded and Real-Time Computing Systems and Applications, 2006. Proceedings. 12th IEEE International Conference on*. 3–9. DOI:http://dx.doi.org/10.1109/RTCSA.2006.54

Yue Lu, T. Nolte, I Bate, and L. Cucu-Grosjean. 2012. A Statistical Response-Time Analysis of Real-Time Embedded Systems. In *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd.* 351–362. DOI:http://dx.doi.org/10.1109/RTSS.2012.85

Mohamed Marouf and Yves Sorel. 2011. Scheduling non-preemptive hard real-time tasks with strict periods. In *Emerging Technologies Factory Automation (ETFA), 2011 IEEE 16th Conference on*. IEEE, 1–8.

Dorin Maxim, Olivier Buffet, Luca Santinelli, Liliana Cucu-Grosjean, and Robert I Davis. 2011. Optimal Priority Assignment Algorithms for Probabilistic Real-Time Systems.. In *RTNS*. Citeseer, 129–138.

Dorin. Maxim and L. Cucu-Grosjean. 2013. Response Time Analysis for Fixed-Priority Tasks with Multiple Probabilistic Parameters. In *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*. 224–235. DOI:http://dx.doi.org/10.1109/RTSS.2013.30

Dorin Maxim, Mike Houston, Luca Santinelli, Guillem Bernat, Robert I. Davis, and Liliana Cucu-Grosjean. 2012. Re-sampling for Statistical Timing Analysis of Real-time Systems. In *Proceedings of the 20th International Conference on Real-Time and Network Systems (RTNS '12)*. ACM, New York, NY, USA, 111–120. DOI:http://dx.doi.org/10.1145/2392987.2393001

Mitra Nasri, Gerhard Fohler, and Mehdi Kargahi. 2014. A Framework to Construct Customized Harmonic Periods for Real-Time Systems. In *Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on*. IEEE, 211–220.

S. Nassif, K. Bernstein, D.J. Frank, A. Gattiker, W. Haensch, B.L. Ji, E. Nowak, D. Pearson, and N.J. Rohrer. 2007. High Performance CMOS Variability in the 65nm Regime and Beyond. In *Electron Devices Meeting, 2007. IEDM 2007. IEEE International*. 569–571. DOI:http://dx.doi.org/10.1109/IEDM.2007.4419002

L. Sha, R. Rajkumar, and J. P. Lehoczky. 1990. Priority inheritance protocols: an approach to real-time synchronization. *IEEE Trans. Comput.* 39, 9 (Sep 1990), 1175–1185. DOI:http://dx.doi.org/10.1109/12.57058

T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J. W S Liu. 1995. Probabilistic performance guarantee for real-time tasks with varying computation times. In *Real-Time Technology and Applications Symposium, 1995. Proceedings*. 164–173. DOI:http://dx.doi.org/10.1109/RTTAS.1995.516213

Tianyi Wang, Qiushi Han, Shi Sha, Wujie Wen, Gang Quan, and Meikang Qiu. 2016. On harmonic fixed-priority scheduling of periodic real-time tasks with constrained deadlines. In *Proceedings of the 53rd Annual Design Automation Conference*. ACM, 131.

Tianyi Wang, Linwei Niu, Shaolei Ren, and Gang Quan. 2015. Multi-core fixed-priority scheduling of real-time tasks with statistical deadline guarantee. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*. EDA Consortium, 1335–1340.

Received Month Year; revised Month Year; accepted Month Year