Multi-Core Fixed-Priority Scheduling of Real-Time Tasks with Statistical Deadline Guarantee

Tianyi Wang[1], Linwei Niu[2], Shaolei Ren[1], and Gang Quan[1]
[1]Department of Electrical&Computer Engineering, Florida International University, FL USA
[2]Department of Mathematics&Computer Science, West Virginia State University, WV USA
E-mail[1]: {tiawang, sren, gaquan}@fiu.edu, E-mail[2]: lniu@wvstateu.edu

***Abstract*—The rising performance variance of IC chips and increased resource sharing in multi-core platforms have significantly degraded the predictability of real-time systems. The traditional deterministic approaches can be extremely pessimistic, if not infeasible at all. In this paper, we adopt a probabilistic approach for fixed-priority preemptive scheduling of real-time tasks on multi-core platforms with statistical deadline miss probability guarantee. Rather than a single-valued worst-case execution time (WCET), we formulate the task execution time as a probabilistic distribution. We develop a novel algorithm to partition real-time tasks on multiple homogenous cores, which takes not only task execution time distributions but their period relationships into considerations. Our extensive experimental results show that our proposed methods can greatly improve the schedulability of real-time tasks when compared with the traditional bin packing approaches.**

***Index Terms*—probabilistic; multi-core; task partitions; harmonic; real-time systems**

I. Introduction

With the fast pace of technology scaling, the performance of IC chips becomes less and less deterministic. Process variation and environmental variation change the performance of IC chips (e.g., maximum clock frequency, power consumption, and etc). Frequency variation can be as much as 30% and up to 20x variation in chip leakage power consumption for a processor designed in 180nm technology [1]. When temperature changes from $20°C$ to $60°C$, as much as 10% variation in dynamic power and 14x variation in leakage power are measured for an ARM Cortex M3 processor [2]. Moreover, multi-core becoming mainstream leads to increased sharing on multi-core platforms which make program executions less predictable. Such indeterminism 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 [3], [4], [5] such that all jobs from every single task can all meet their deadlines. As the computing performance becomes less and less predictable, such a deterministic design can lead to extremely pessimistic design. In addition, the hard deadline guarantee may not be necessary for many soft real-time systems that allow a portion of the jobs miss their deadlines. For example, for aerospace industry, a probability of failure of 10^{-15} per hour is considered to be feasible compared to the maximum allowed probability of failure of 10^{-9} per hour that is required by the certification authorities [6].

This work is supported in part by NSF under projects CNS-1423137 and CNS-1018108.

The probabilistic approach takes system probabilistic characteristics, such as the execution times, into account for real-time system analysis and design to prevent over-provisioning and, at the same time, meet real-time constraints [7]. There have been increasing interests from real-time community on probabilistic approaches for real-time system analysis and design. For example, Tia et al. [8] presented a probabilistic performance guarantee for semi-periodic tasks by transforming semi-periodic tasks into a periodic task followed by a sporadic task. Atlas et al. [9] introduced a statistical rate monotonic scheduling for periodic tasks with statistical QoS requirements. Maxim et al. [10] 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 response time distributions in order to ease timing analysis for real-time systems in [11]. Yue et al. [12] presented a statistical response time analysis by analysing samples in timing traces taken from real systems. In [13], 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 [14] 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 [15], the authors proposed a new convolution-based stochastic analysis in which they modeled faults as additional execution time to bound the probability to exceed a response-time value in the worst-case on single processor under fixed-priority non-preemptive scheduling policy.

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 [16]. Given the NP-hard nature of this problem [17], one intuitive approach is to transform the problem into a simple bin-packing problem [18], and employ the feasibility test methods developed in [13] 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 [19], [20]. The challenge however is how to determine if a task is more "harmonic" than another one to a

978-3-9815370-4-8/DATE15/©2015 EDAA

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 traditional bin-packing approaches.

The rest of the paper is organized as follows. In Section II we introduce our system models and formally define our problem. In section III we present the harmonic-aware metrics we developed. In Section IV we talk about our task partition algorithm in details. Section V presents the experimental results and finally we conclude in Section VI.

II. Preliminary

In this section we first introduce our system models such as task models and processor models. And then we formulate the problem formally.

A. 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 Rate Monotonic Scheduling (RMS) policy. Each task $\tau_i \in \Gamma$, characterized by a tuple (C_i, T_i), where

$$C_i = \begin{pmatrix} c_1 = c_{min} & ...c_k & \cdots & c_n = c_{max} \\ Pr(c_{min}) & ...Pr(c_k) & \cdots & Pr(c_{max}) \end{pmatrix} \quad (1)$$

representing the execution time distribution of τ_i. That is, the probability that the execution time of $C_i = c_k$ is $Pr(c_k)$. For all possible values of C_i, we have $c_k \in [c_{min}, c_{max}]$, where c_{min} and c_{max} are the minimum and maximum values for C_i, and $\sum_{k=1}^{n} Pr(c_k) = 1$. T_i is the period of task τ_i which is a constant value. We assume deadline equals to period in this paper, $D_i = T_i$.

Since a task's execution time is not unique, the response time for each job may be different. Therefore sometimes 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:*

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

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) can be formulated as

$$DMP_i = max\{DMP_{i,j}\}, \tau_{i,j} \in \tau_i, \quad (3)$$

and the deadline miss probability for a task set Γ (denoted as DMP_Γ) can be formulated as

$$DMP_\Gamma = max\{DMP_i\}, \tau_i \in \Gamma. \quad (4)$$

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 processing cores, $\mathcal{P} = \{p_1, p_2, ...p_K\}$,*
- *and the deadline miss probability, DMP_Γ,*

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 of the task set is satisfied and the number of cores is minimized.

B. Motivations

One simple approach for this problem is to transform it into the traditional bin-packing problem. Note that, with the knowledge of tasks assigned to a processing core, Lopez et al. [21] proposed a method to calculate the probabilistic response time of a job under a preemptive uniprocessor fixed-priority scheduling policy, which can be further applied to determine if the DMP constraints can be satisfied. Therefore, traditional bin-packing approaches such as First Fit, Next Fit, Best Fit can be 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 the tasks are harmonic, i.e. task periods are integer multiples of each other. For tasks that are not entirely harmonic, Fan et al. [19] showed that, if the period relationships among tasks can be appropriately exploited, the processor utilization can be significantly improved. Specifically, they introduce three interesting concepts, *sub harmonic task set*, *the primary harmonic task set* and *harmonic index*, which are defined as follows:

Definition 2. *[19] 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 $T_i < T_j$ ($a|b$ 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. *[19] 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. *[19] 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*

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

where

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

$U(\Gamma)$ and $U(\Gamma')$ represent the utilizations of task set Γ and Γ', respectively. We can employ *Sr* or *DCT* algorithm [22], [23] to find all the sub harmonic task sets with a complexity as low as $N \cdot log(N)$.

When allocating tasks to processors, they either allocate a task to a processor that can minimize the harmonic index, or pick the sub task sets with highest degree of harmonic and assign to a processor.

We believe that, by exploiting the period relationships among tasks, we can also greatly improve the processor utilization in Problem 1. The challenge is how to quantify the degree of harmonic among different tasks with probabilistic execution times. For tasks with deterministic execution times, according to Definition 4, for a given reference task, tasks with its original period closer to that

in its primary harmonic task set has a high degree of harmonic. However, the degree of harmonic of a task to its reference task may depend not only on its period but 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. If we combine τ_a and τ_b, we have $DMP_{\tau_a,\tau_b} = 0$. If we combine τ_a and τ_c, we have $DMP_{\tau_a,\tau_c} = 24.5\%$. Therefore, the degree of harmonic of a task depends not only on its period, but its execution time distribution as well.

In what follows, we first introduce four metrics that we have developed, with each one improving upon the previous, 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 considerations to guide the partition process for periodic tasks with random execution times on multi-core platforms.

III. Harmonic index for tasks with probabilistic execution times

In this section, we formally introduce the metrics which take harmonic relationships 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 *harmonic* of tasks and allocate tasks with high degree of harmonic to the same processor to achieve high utilization as well as high schedulability. In what follows, we develop four metrics to measure the task harmonic relationships.

A. 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 $\tau_r \in \Gamma$ and $\tau'_r \in \Gamma'$ 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 also 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 be therefore defined in the 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

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

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

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

TABLE I
Sub harmonic task set transformations of a 4-task set

			Transformed based on τ_1 $(T_1 \mid T_i)$		Transformed based on τ_2 $(T_2 \mid T_i)$	
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

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 [23]). 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 $\mathcal{H}_m(\tau_2, \tau_1) = 0.3$. Then we tentatively combine task τ_1 with τ_3, task τ_1 with τ_4 and task τ_1 with τ_2 into a sub set, to check deadline miss probability of each individual sub set. We can get: $DMP_{\tau_1,\tau_3} = 0\%$, $DMP_{\tau_1,\tau_4} = 19.55\%$ and $DMP_{\tau_1,\tau_2} = 24.5\%$. It 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 ($\mathcal{H}_m$) quantifies the harmonic relationship of a task to its reference task by measuring the difference of its expected utilization with that in the primary harmonic task set. While mean value is a good representative value for a probabilistic distribution, it cannot capture the entire characteristics of a probabilistic distribution. Let us recall the example shown in Sub-section II-B, task τ_b and τ_c 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 ($DMP_{\tau_a,\tau_b} \neq DMP_{\tau_a,\tau_c}$). More effective harmonic index needs to be developed.

B. Variance based harmonic index

From example in the previous sub-section, we can observe that the harmonic index depends not only on the mean value of the execution times, but also the variance of the execution times as well. 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 $\tau'_i = \{C_i, T'_i\}$ 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

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

$\mathcal{H}_v$ improves upon $\mathcal{H}_m$ by taking both the mean value of execution times as well as their variance into considerations. For the example shown in Sub-section II-B, 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. 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 harmonic. 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.

C. 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 $CDF_{\tau_i}(x)$, can be formulated as*

$$CDF_{\tau_i}(x)=\frac{Pr(C_i\le x)}{T_i} \tag{12}$$

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 $x=[x_1,x_2,...,x_n]$, its ℓ^2-norm, denoted as $||x||$, is defined as*

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

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 $\Gamma'(\tau_r)$. Then the* Cumulative distribution function harmonic index *of task τ_i to τ_r, denoted as $\mathcal{H}_C(\tau_i,\tau_r)$, is defined as*

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

where x represents sampling point for the two cumulative distribution functions, so that we can apply the above equation to measure harmonic index.

One thing to note is that for $CDF_{\tau_i}(x)$ and $CDF'_{\tau_i}(x)$, they may have different sampling points. For this case, we enumerate all the sampling points for both CDFs to calculate $\mathcal{H}_C(\tau_i,\tau_r)$.

The rationale behind the definition of *Cumulative distribution function harmonic index* is that we intend to determine the degree of harmonic by measuring how different the task utilization distribution has changed after changing its period to be integer multiple of that for the reference task. The larger the difference, the less harmonic the two tasks are.

As indicated in the equation, the *harmonic index* tries to evaluate the closeness between two distributions by summing up the square difference of each sampling point of the two distributions (CDFs) and then take the root value as the final result. Consider the same example in Table I. After all the tasks have been transformed based on the *reference task* τ_1's period, we calculate the *harmonic index* between τ_i and τ_1, i.e., $\mathcal{H}_C(\tau_i,\tau_1)$. In this way, we can rank the harmonic relationships of all the tasks compared to the *reference task* τ_1 and find out which tasks are more harmonic to task τ_1.

D. The utilization sum based harmonic index

Note that $\mathcal{H}_C$ 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 individually.

For ease of our presentation, we first introduce the following notation. Let $\tau_i\bigotimes\tau_r$ denote the convolution of $\mathcal{C}_{\tau_i}$ and $\mathcal{C}_{\tau_r}$.

Definition 11. *Given a task τ_i and a* reference task *τ_r, let CDFs for $\tau_i\bigotimes\tau_r$ and $\hat{\tau}_i\bigotimes\tau_r$ be CDF_{τ_i,τ_r} and $CDF_{\hat{\tau}_i,\tau_r}$. Then the* Utilization sum based harmonic index *of $\tau_i\bigotimes\tau_r$ and $\hat{\tau}_i\bigotimes\tau_r$, denoted as $\mathcal{H}_S(\tau_i,\tau_r)$, is formulated in Equation 15,*

$$\mathcal{H}_S(\tau_i,\tau_r)=||CDF_{\tau_i,\tau_r}(x)-CDF_{\hat{\tau}_i,\tau_r}(x)|| \tag{15}$$

$\mathcal{H}_S$ can take the information of combined utilization distribution CDF_{τ_i,τ_r} between Γ and $\Gamma'(\tau_r)$ into account, therefore it can produce better results in terms of harmonic relationships.

So far we have introduced all four metrics with each improving upon another. These metrics are critical for our task partition algorithm to make mapping decisions. In what follows, we present our task partition algorithm in details.

IV. Task partitioning algorithm

With the harmonic indexes defined above, we are ready to introduce our task partition algorithm. Essentially, our algorithm intends to identify the tasks with highest harmonic index values, and put them into one processor to improve the processor utilization. To satisfy the DMP requirement, we conduct the schedulability analysis based on the technique proposed in [13]. 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 the sub sets from each primary harmonic task set, we choose the best sub-set and allocate them to a processor. Then we delete these tasks from task set Γ. We repeat this process for the rest of the tasks until all tasks are assigned.

We want to explain how we choose the best sub-set by an example. Still we are going to analyze the example shown in Table I. Let

Algorithm 1 Stochastic task partitioning algorithm.

Input:

1: 1) Task set: $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$;
2: 2) Task set deadline miss probability: DMP_Γ;

Output:

3: Task partitions: $= \{subset_1, subset_2, \ldots, subset_K\}$, K is the total number of processors.
4: **while** $\Gamma \neq \emptyset$ **do**
5: $SubSet = \emptyset$; //initialize the sub-set to empty
6: $Probability = 0$; //initialize the probability of utilization distribution of a sub-set greater than threshold (0.7 in this case) to 0
7: $\Gamma' = \{\Gamma'(\tau_1), \Gamma'(\tau_2), \ldots, \Gamma'(\tau_L)\}$, // identify all the primary harmonic task sets, where L is the total number of primary harmonic task sets. If no two tasks such that their periods can satisfy $T_i | T_j$ $(i < j)$, then $L = N$, otherwise $L < N$;
8: **for** $i = 1$ to L // for each primary harmonic task set **do**
9: $\Gamma'(\tau_i) = \{\tau_1^{'}, \tau_2^{'}, \ldots, \tau_N^{'}\}$ // sort the tasks with increasing order of $\mathcal{H}_m$, $\mathcal{H}_v$, $\mathcal{H}_C$ or $\mathcal{H}_S$
10: $subset_i = \emptyset$ // initialize a sub-set for $\Gamma'(\tau_i)$
11: **for** $j = 1$ to N **do**
12: $subset_i = subset_i + \tau_j^{'}$
13: **if** $DMP_{subset_i} > DMP_\Gamma$ **then**
14: $subset_i = subset_i - \tau_j^{'}$;
15: break;
16: **end if**
17: **end for**
18: **if** $Pr(\mathcal{U}_{subset_i} > 0.7) > Probability$ **then**
19: $Probability = Pr(\mathcal{U}_{subset_i} > 0.7)$;
20: $SubSet = subset_i$;
21: **end if**
22: **end for**
23: $\Gamma = \Gamma - SubSet$;
24: **end while**

us take task τ_1 as reference task, then the two corresponding sub-sets: (τ_1, τ_3) and (τ_2, τ_4), respectively, are both perfectly schedulable. Now the question is which one should we pick first in order to generate better sub-set? The criteria here is that we want to utilize the processors as much as possible, therefore, we would like to pick the sub-set with large utilization. However, since the tasks we are discussing in this paper have random execution times, how should we define which sub-set has larger utilization than another?

From the task's execution time distribution, we can easily get each task's utilization distribution. Let $\mathcal{U}_i$ denote the utilization distribution for task τ_i. For the tasks in Table I we have $\mathcal{U}_1 = \begin{pmatrix} \frac{2}{6} & \frac{3}{6} \\ 0.3 & 0.7 \end{pmatrix}$, $\mathcal{U}_2 = \begin{pmatrix} \frac{4}{10} & \frac{5}{10} \\ 0.5 & 0.5 \end{pmatrix}$, $\mathcal{U}_3 = \begin{pmatrix} \frac{4}{12} & \frac{6}{12} \\ 0.5 & 0.5 \end{pmatrix}$, and $\mathcal{U}_4 = \begin{pmatrix} \frac{8}{20} & \frac{10}{20} \\ 0.7 & 0.3 \end{pmatrix}$. The utilization distribution of a sub-set is the convolution of each task within the sub-set. So we have $\mathcal{U}_{1,3} = \mathcal{U}_1 \bigotimes \mathcal{U}_3 = \begin{pmatrix} 0.67 & 0.83 & 1.0 \\ 0.15 & 0.5 & 0.35 \end{pmatrix}$, similarly we have $\mathcal{U}_{2,4} = \begin{pmatrix} 0.8 & 0.9 & 1.0 \\ 0.35 & 0.5 & 0.135 \end{pmatrix}$ (transferred to decimal for better illustration). If we choose the utilization threshold as 0.8, we can calculate that the probability that sub-set (τ_1, τ_3) has utilization distribution greater than 0.8 is $Pr(\mathcal{U}_{1,3} > 0.8) = 0.85$ while that for sub-set (τ_2, τ_4)'s under the same situation is $Pr(\mathcal{U}_{2,4} > 0.8) = 1.0$. So we will choose sub-set (τ_2, τ_4) first because it has higher probability to have larger utilization than sub-set (τ_1, τ_3). The threshold we choose in this paper is 0.7 (higher threshold may result in a small improvement of partitioning but more computational expense).

V. Experimental results

In this section, we use experiments to investigate the effectiveness of our proposed algorithm. Two sets of experiments are conducted. First, we compared the performance in terms of number of cores necessary with different partition algorithms. Second, we compared the success probabilities by different approaches when scheduling real-time tasks on a multi-core platform with a pre-defined core number.

A. Experiment setup

In our experiments, we randomly generated the stochastic tasks. Specifically, for each task we generated four different possible execution times and the corresponding probabilities (the sum of the four probabilities equals to 1). We did not make any assumption regarding task's execution time distribution, therefore any distribution can be applied. We generated the periods for all the tasks in a way that the expected utilization of each task is evenly distributed within $[0.2, 0.5]$.

Five different approaches were realized in our experiment, i.e., four task partitioning algorithms with the four proposed harmonic indexes and one traditional bin packing approach, first fit.

We denote our approach using *mean based harmonic index* as H_m, using *variance based harmonic index* as H_v, the one using *cumulative distribution based harmonic index* as H_{cdf} and the last one using *distribution sum based harmonic index* as H_{sum}. Then we compare the four approaches with first fit (FF) algorithm.

Specifically, for FF approach, we sort the tasks according to their expected utilizations and pack as many tasks as possible from the top of the task queue one by one, until we can form a sub-set while meeting DMP_Γ constraint. After we successfully find a sub-set, we delete those tasks from the task set and repeat this process until all the tasks have been partitioned.

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

In this experiment, we study the performance differences in terms of number of processors used by different approaches when scheduling given task sets. Three different test cases were generated and tested: 8 tasks, 16 tasks and 24 tasks. For each test case, we randomly generated 50 task sets and set $DMP_\Gamma = 10\%$. The results are shown in Figure 1.

From Figure 1, we can see that FF algorithm always utilize more processors than all the other approaches. This is because it does not consider the harmonic relationships between tasks and therefore wastes processor resources. All four other approaches, by taking the task harmonic relationship into consideration, outperform FF significantly. It is interesting to see that the performance improvement increases following the order of H_m, H_v, H_{cdf} and H_{sum}.

The first two approaches, H_m and H_v have low computational overhead. However, they cannot accurately capture the harmonic relationships between tasks since they focus only on the expected values and variances of the tasks. The other two approaches, on the other hand, are more elaborative and have higher computational cost. However, they determine the harmonic relationship based on the entire distribution of a task and therefore can result in better mappings. As a results, we can see that, when the task number is 16, the latter two approaches in average can save one core in

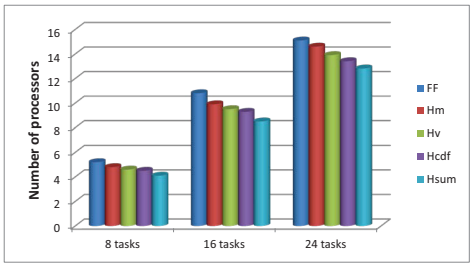

Fig. 1. Processor usage versus task number with $DMP_\Gamma = 10\%$

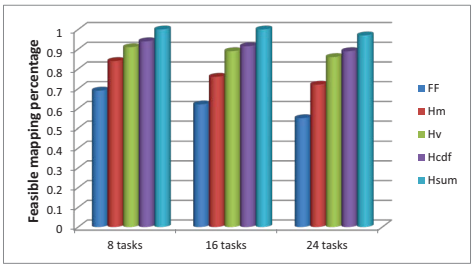

Fig. 2. Feasible mapping percentages for different approaches

scheduling the same task set. Moreover, with the increase of the task number, the flexibility to allocate tasks increases. Therefore, the performance improvement increases. For example, for 8 tasks, the average improvement is about 0.5 core savings and for 24 tasks, the average improvement becomes about 1.4 core savings.

C. Performance w.r.t. schedulability

Next, we analyze the performance of different approaches in terms of schedulability. That is, for a given core number and task sets, how many task sets can be successfully scheduled. We used the same three test cases for the first experiment and set the core number to 5, 10 and 14 for 8 tasks, 16 tasks and 24 tasks, respectively. We show the results in Figure 2. Similar conclusions can be drawn from Figure 2 and H_{sum} has the highest scheduling rates among all five approaches. For 8 tasks, H_{sum} can improve upon FF by 30% and H_m by 16%. Also, with the increase of core numbers, the flexibility increases and therefore the performance is better. For example when there are 24 tasks in the task set, H_{sum} can improve upon FF by 45% and H_m by 28%.

VI. 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 of real-time systems. In this paper, we develop 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 indexes to quantify the harmonic among tasks, and based on which to better identify task set allocations and improve processor utilization. We conducted extensive simulation study and the results show that our algorithms can significantly outperform the existing approach.

References

[1] S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De, "Parameter variations and impact on circuits and microarchitecture," in *Proceedings of the 40th annual Design Automation Conference*, 2003, pp. 338–342.

[2] L. Wanner, C. Apte, R. Balani, P. Gupta, and M. Srivastava, "Hardware variability-aware duty cycling for embedded sensors," vol. PP, no. 99, 2012, pp. 1–1.

[3] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973. [Online]. Available: http://doi.acm.org/10.1145/321738.321743

[4] J. Lehoczky, L. Sha, and Y. Ding, "The rate monotonic scheduling algorithm: exact characterization and average case behavior," in *Real Time Systems Symposium, 1989., Proceedings.*, Dec 1989, pp. 166–171.

[5] J. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Real-Time Systems Symposium, 1990. Proceedings., 11th*, Dec 1990, pp. 201–209.

[6] A. 653, "An avionics standard for safe, partitioned systems," in *Wind River Systems/IEEE Seminar*, 2008.

[7] S. Edgar and A. Burns, "Statistical analysis of wcet for scheduling," in *Real-Time Systems Symposium, 2001. (RTSS 2001). Proceedings. 22nd IEEE*, Dec 2001, pp. 215–224.

[8] T.-S. Tia, Z. Deng, M. Shankar, M. Storch, J. Sun, L.-C. Wu, and J. W. S. Liu, "Probabilistic performance guarantee for real-time tasks with varying computation times," in *Real-Time Technology and Applications Symposium, 1995. Proceedings*, May 1995, pp. 164–173.

[9] A. Atlas and A. Bestavros, "Statistical rate monotonic scheduling," in *Real-Time Systems Symposium, 1998. Proceedings., The 19th IEEE*, Dec 1998, pp. 123–132.

[10] D. Maxim, O. Buffet, L. Santinelli, L. Cucu-Grosjean, and R. I. Davis, "Optimal priority assignment algorithms for probabilistic real-time systems." in *RTNS*. Citeseer, 2011, pp. 129–138.

[11] D. Maxim, M. Houston, L. Santinelli, G. Bernat, R. I. Davis, and L. Cucu-Grosjean, "Re-sampling for statistical timing analysis of real-time systems," in *Proceedings of the 20th International Conference on Real-Time and Network Systems*, ser. RTNS '12. New York, NY, USA: ACM, 2012, pp. 111–120.

[12] Y. Lu, T. Nolte, I. Bate, and L. Cucu-Grosjean, "A statistical response-time analysis of real-time embedded systems," in *Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd*, Dec 2012, pp. 351–362.

[13] K. Kim, J. Diaz, L. Lo Bello, J. Lopez, C.-G. Lee, and S.-L. Min, "An exact stochastic analysis of priority-driven periodic real-time systems and its approximations," *Computers, IEEE Transactions on*, vol. 54, no. 11, pp. 1460–1466, Nov 2005.

[14] D. Maxim and L. Cucu-Grosjean, "Response time analysis for fixed-priority tasks with multiple probabilistic parameters," in *Real-Time Systems Symposium (RTSS), 2013 IEEE 34th*, Dec 2013, pp. 224–235.

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

[16] E. Bini and G. C. Buttazzo, "Measuring the performance of schedulability tests," *Real-Time Syst.*, vol. 30, no. 1-2, pp. 129–154, May 2005.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman & Co., 1990.

[18] D. Johnson, A. Demers, J. Ullman, M. Garey, and R. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," *SIAM Journal on Computing*, vol. 3, no. 4, pp. 299–325, 1974.

[19] M. Fan and G. Quan, "Harmonic semi-partitioned scheduling for fixed-priority real-time tasks on multi-core platform," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, March 2012, pp. 503–508.

[20] ——, "Harmonic-aware multi-core scheduling for fixed-priority real-time systems," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 6, pp. 1476–1488, June 2014.

[21] J. Lpez, J. Daz, J. Entrialgo, and D. Garca, "Stochastic analysis of real-time systems under preemptive priority-driven scheduling," *Real-Time Systems*, vol. 40, no. 2, pp. 180–207, 2008.

[22] C.-C. Han, K.-J. Lin, and C.-J. Hou, "Distance-constrained scheduling and its applications to real-time systems," *IEEE Trans. Comput.*, vol. 45, no. 7, pp. 814–826, Jul. 1996. [Online]. Available: http://dx.doi.org/10.1109/12.508320

[23] C.-C. Han and H. ying Tyan, "A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms," in *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, Dec 1997, pp. 36–45.