Best-Harmonically-Fit Periodic Task Assignment Algorithm on Multiple Periodic Resources

Chunhui Guo, *Student Member, IEEE*, Xiayu Hua, *Student Member, IEEE*, Hao Wu, *Student Member, IEEE*, Douglas Lautner, *Student Member, IEEE*, Shangping Ren, *Senior Member,IEEE*

Abstract—The periodic task set assignment problem in the context of multiple processors has been studied for decades. Different heuristic approaches have been proposed, such as the Best-Fit (BF), the First-Fit (FF), and the Worst-Fit (WF) task assignment algorithms. However, when processors are not dedicated but only periodically available to the task set, whether existing approaches still provide good performance or if there is a better task assignment approach in the new context are research problems which, to our best knowledge, have not been studied by the real-time research community. In this paper, we present the Best-Harmonically-Fit (BHF) task assignment algorithm to assigning periodic tasks on multiple periodic resources. By periodic resource we mean that for every fixed time interval, i.e., the period, the resource always provides the same amount of processing capacity to a given task set. Our formal analysis indicates that if a harmonic task set is also harmonic with a resource's period, the resource capacity can be fully utilized by the task set. Based on this analysis, we present the Best-Harmonically-Fit (BHF) task assignment algorithm. The experimental results show that, on average, the BHF algorithm results in 53.26%, 42.54%, and 27.79% higher resource utilization rate than the Best-Fit Decreasing (BFD), the First-Fit Decreasing (FFD), and the Worst-Fit Decreasing (WFD) task assignment algorithms, respectively; but comparing to the optimal resource utilization rate found by exhaustive search, it is about 11.63% lower.

Index Terms—Real-Time Systems; Task Assignment; Periodic Resource; Best-Harmonically-Fit

1 Introduction

As computer hardware technology advances, both the number of processing units and the computational capabilities of these processing units used in real-time systems are increasing. To take the advantages of the increased physical resource's capacity, multiple groups of real-time applications are deployed on the same physical platform. However, because each group of real-time applications may have different time granularity [1], when multiple groups of applications share the same processors, traditional real-time system models and theoretic results may not be sufficient or even applicable to guarantee that each group of real-time applications will not have time interference among each other and that they will satisfy their timing requirements.

In order to study the issues of scheduling multiple groups of real-time applications on the same physical resources, the concept of virtual real-time resources is proposed [1]. Virtual real-time resources are an abstraction of physical resources where the physical resources are shared by real-time application groups [1]. With the concept of virtual real-time resources, each group of real-time applications has its own isolated and independent virtual resource, hence avoiding interference among different groups of real-time applications. However, from an application perspective, virtual real-time resources are not continuously available to the application. Instead, virtual resources are periodic, i.e., they periodically provide certain amount of processing capability to the applications [1], [2], [3], [4], [5].

The authors are with Computer Science Department, Illinois Institute of Technology, Chicago, IL 60616, USA
Emails: {cguo13, xhua, hwu28, dlautner}@hawk.iit.edu, ren@iit.edu

The study of periodic resources can be traced back to 1999 when the concept of periodic resource was first formally defined [4]. It has recently drawn more attention in the community [1], [5], [6], [7], [8], [9], [10], [11], [12]. However, until now, most of the studies about periodic resources focus on task set schedulability analysis on a *single* periodic resource. There has not been much work, if any, in the literature dealing with the task assignment problem on multiple periodic resources in computer cloud. In this paper, "multiple resources" refers to multiple processing units. Under a computer cloud environment, we have virtually unlimited resources where a task set can be scheduled on, the problem is how to minimize the total resource usage, i.e., maximize resource utilization rate and minimize the number of resources used. The paper studies the task assignment problem in the context of assigning multiple periodic tasks to multiple periodic processing units. The main goal is to decide on a task assignment strategy that maximally utilizes the resource capacities provided by periodic resources.

To achieve the goal, we first study the period relationship between a task and a resource that enables the resource capacity be fully utilized by the task. Intuitively, the more harmonically related the task and the periodic resource are, the better resource utilization rate we can achieve. We formally prove that, in fact, if a harmonic task set is also harmonic with a periodic resource, the resource capacity can be fully (100%) utilized under the RM (rate-monotonic) scheduling algorithm. Second, based on the harmonicity property, we present the Best-Harmonically-Fit (BHF) task assignment algorithm to assign a periodic task set to multiple periodic resources. We then experimentally evaluate the BHF algorithm's performance by comparing it with commonly used multiprocessor task assignment approaches in the literature, i.e., the Best-Fit Decreasing, the First-Fit Decreasing, and the Worst-Fit Decreasing approaches [13], [14], [15], [16]. We also evaluate the BHF algorithm by comparing it with the optimal task assignment found through exhaustive search for a small-sized task set and resource set.

The rest of the paper is organized as follow: Section 2 discusses related work. Section 3 defines system models, presents preliminary results, and formulates the problem we are to address. Section 4 presents the harmonic utilization bound for a single periodic resource and a task set harmonic transformation with respect to a periodic resource. In Section 5, we introduce the Best-Harmonically-Fit (BHF) task assignment algorithm. Section 6 discusses experimental results. We conclude and point out future work in Section 7.

2 Related Work

Since the rate-monotonic (RM) and the earliest deadline first (EDF) scheduling algorithms were first analyzed by Liu and Layland in 1973 [17], the real-time scheduling problem has been studied extensively. To this day, the rate-monotonic scheduling algorithm is still considered the most significant fixed-priority scheduling algorithm for scheduling periodic tasks on a single dedicated resource. The RM utilization bound for a preemptive system is $N(2^{1/N}-1)$ with its limit of 69.3% on a single dedicated resource, where N is the number of tasks [17]. Mossé *et al.* [18], [19] proposed an R-Bound based on the Liu and Layland bound. The R-Bound takes task period differences into consideration and tightens the utilization bound to $(N-1)(r^{1/(N-1)}-1)+2/r-1$ with limit $\ln r+2/r-1$, where $r=T_{\max}/T_{\min}$ and $1 \leq r < 2$. Another improvement on the RM bound was made by Han *et al.* [20]. They proved that the utilization bound can reach 100% with the RM scheduling algorithm if the task set has harmonic periods.

Another major research area in the real-time community is scheduling tasks on multiprocessors. The goal of multiprocessor scheduling is to schedule as many tasks as possible (from the total task utilization perspective) on a given number of processors while still guaranteeing task set deadline satisfication [21], [15], [22] or minimize the number of used processors to guarantee task set deadlines [23]. However, as stated by Liu and Layland in [17], scheduling periodic tasks on multiprocessors is much harder than on a single processor. The utilization bound for a multiprocessor system is much lower than for a single processor. Many algorithms are proposed to improve the utilization bound for multiprocessors. However, it is proven that the optimal utilization bound for a multiprocessor scheduling algorithm is only $(M+1)/2$ [21], where M is the number of processors. Recently, researchers have improved the utilization bound for multiprocessors under certain conditions. Wang *et al.* proved that the utilization bound for multiprocessors can reach Liu and Layland bound for a single processor if tasks can be split [15]. Fan and Quan proposed a harmonic-aware scheduling approach to improve schedulability on multiprocessors [24], [25], [22].

However, the literature mentioned above is based on the assumption that resources are dedicated resources, i.e., they are constantly available to application tasks. When multiple groups of real-time applications share the same physical resources, the assumption that resources are constantly available to the application becomes invalid. Hence, the concept of virtual real-time resources is proposed to handle such scenario. A virtual resource is often represented as $\gamma=(\Pi,\Theta)$, where Π is the virtual resource period and Θ is the processing time available to applications [1], [2], [3], [4], [5].

Shirero *et al.* [4] first defined periodic resources and proposed a real-time round robin scheduling algorithm in 1999. They also introduced the concept of resource regularity. Based on resource regularity, they gave a schedulability bound for periodic tasks on a single periodic resource. Mok *et al.* [5], [26] extended Shirero's work and proposed a more comprehensive schedulability analysis for periodic resources under both EDF and RM scheduling algorithms. However, both Shirero's and Mok's periodic resource models have a constraint that either the resource available pattern within each period or the resource regularity is known and does not change at run-time. Shin *et al.* later removed the constraint on the periodic resource model and provided the schedulability analysis under a relaxed model where resource pattern can be arbitrary and can change at run-time. They gave schedulability bounds under the relaxed model for both EDF and RM [6], [7], [8], [9].

Hua *et al.* [12] recently studied how multiple periodic resources may be integrated into an equivalent single periodic resource so that existing real-time scheduling theorems on a single periodic resource can be applied. They further extended schedulability tests of periodic tasks on a single periodic resource from the continuous time domain given in [9] to a discrete time domain so that the schedulability tests given in [9] can be applied in practice.

However, as of today, much of the work in the literature has been on schedulability analysis of a task set on a single periodic resource. To the best of our knowledge, there is no prior work studying the task assignment problem on multiple periodic resources. In this paper, we are to address the problem: for a given set of periodic tasks and a given set of periodic resources, if the task set is schedulable on any single periodic resource in the resource pool, how to assign tasks to the available periodic resources so that the utilization rate of periodic resources used is maximized. The problem we are to address is similar to the work of assigning a periodic task set to processors so as to minimize the number of processors used [23], but it is in a different context where resources are not dedicated processors, but rather periodic in nature.

3 Problem Formulation

3.1 Models and Definitions

Task Model

The task model considered in this paper is similar to the one defined in [17]. In particular, a task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ has N independent periodic tasks that are all released at time 0. Each task $\tau_i \in \Gamma$ is a 2-tuple (T_i, C_i), where T_i is the *inter-arrival time* between any two consecutive jobs of τ_i (also called *period*), and C_i is the *worst-case execution time*. We further assume that a task period is an integer, i.e., $T_i \in \mathbb{N}^+$. The *utilization* of each task τ_i is defined as $U_{\tau_i} = C_i/T_i$, and the *utilization* of the task set Γ is denoted as U_Γ, where

$$U_\Gamma = \sum_{\tau_i \in \Gamma} U_{\tau_i}. \tag{1}$$

We use $U_{\max}$ and $T_{\min}$ to denote the maximum task utilization and the minimum task period of the task set Γ, respectively, i.e.,

$$U_{\max} = \max\{U_{\tau_i} | \forall \tau_i \in \Gamma\}$$
$$T_{\min} = \min\{T_i | \forall \tau_i \in \Gamma\}.$$

Definition 1. *[Harmonic Tasks] Given two periodic tasks τ_i and τ_j, they are harmonic if one task's period can divide the other task's period, i.e.,*

$$T_i \bmod T_j = 0 \vee T_j \bmod T_i = 0. \tag{2}$$

□

Definition 2. *[Harmonic Task Set] Given a task set Γ, it is a harmonic task set if all tasks in Γ are pairwise harmonic, i.e.,*

$$\forall \tau_i, \tau_j \in \Gamma : T_i \bmod T_j = 0 \vee T_j \bmod T_i = 0. \tag{3}$$

□

Resource Model

The resource model used in the paper is the same as the one defined in [6]. In particular, a resource set $\mathcal{R} = \{\gamma_1, \gamma_2, \ldots, \gamma_M\}$ consists of M periodic resources that are all available at time 0. Each resource $\gamma_j \in \mathcal{R}$ is characterized by a 2-tuple (Π_j, Θ_j), where Π_j is the resource *period*, and Θ_j is the *allocation time*. We further assume that both a resource period and a resource allocation time are integers, i.e., $\Pi_j, \Theta_j \in \mathbb{N}^+$, and satisfy $0 < \Theta_j \leq \Pi_j$. The *capacity* of each resource γ_j is defined as $\mathcal{C}_{\gamma_j} = \Theta_j/\Pi_j$, and the *capacity* of a resource set $\mathcal{R}$ is denoted as $\mathcal{C}_\mathcal{R}$, where

$$\mathcal{C}_\mathcal{R} = \sum_{\gamma_j \in \mathcal{R}} \mathcal{C}_{\gamma_j}. \tag{4}$$

We use $\tau \mapsto \gamma$ to denote that task τ is assigned to periodic resource γ, and refer γ as the *host resource* for task τ. For a resource γ and a task set Γ, we use $\Gamma_\gamma \subseteq \Gamma$ to denote the task subset containing all tasks assigned to resource γ, i.e.,

$$\Gamma_\gamma = \{\tau_i | \tau_i \in \Gamma \wedge \tau_i \mapsto \gamma\}. \tag{5}$$

Similarly, the number of tasks in Γ_γ is denoted as N_γ, and the *utilization* of Γ_γ is denoted as

$$U_{\Gamma_\gamma} = \begin{cases} \sum_{\tau_i \in \Gamma_\gamma} U_{\tau_i} & \texttt{if} \quad \Gamma_\gamma \neq \emptyset \\ 0 & \texttt{if} \quad \Gamma_\gamma = \emptyset \end{cases} \tag{6}$$

A periodic resource γ is called an *unused resource* if there are no tasks assigned to it; otherwise it is called a *used resource*. For a resource set $\mathcal{R}$, we use $\mathcal{R}_{\texttt{used}} \subseteq \mathcal{R}$ to denote the resource subset containing all used resources; $M_{\texttt{used}}$ is the size of $\mathcal{R}_{\texttt{used}}$, i.e., $M_{\texttt{used}} = |\mathcal{R}_{\texttt{used}}|$.

Definition 3. [Resource Utilization Rate ($\texttt{UR}_\gamma$)] *The utilization rate of a periodic resource γ is defined as the ratio between the utilization used by a task set assigned to the resource and the resource's capacity, i.e.,*

$$\texttt{UR}_\gamma = \frac{U_{\Gamma_\gamma}}{\mathcal{C}_\gamma}. \tag{7}$$

□

Definition 4. [Resource Set Utilization Rate ($\texttt{UR}_\mathcal{R}$)] *The utilization rate of a resource set $\mathcal{R}$ is defined as the total capacity percentage used by tasks assigned to $\mathcal{R}$, i.e.,*

$$\texttt{UR}_\mathcal{R} = \frac{\sum_{\gamma_j \in \mathcal{R}_{\texttt{used}}} U_{\Gamma_{\gamma_j}}}{\sum_{\gamma_j \in \mathcal{R}_{\texttt{used}}} \mathcal{C}_{\gamma_j}}. \tag{8}$$

□

3.2 Preliminary Results

For self-containment, we introduce a few terms and schedulability analysis results from [9], and give our corollaries derived from the theorems given in [9].

Theorem 1. *[9] Given a task set Γ and a single periodic resource $\gamma = (\Pi, \Theta)$, if $\forall i, 1 \leq i \leq N, T_i \geq 2\Pi - \Theta$, the task utilization bound under RM scheduling is*

$$\texttt{UB}_\gamma = \mathcal{C}_\gamma \cdot N_\gamma \left[\left(\frac{2k + 2(1 - \mathcal{C}_\gamma)}{k + 2(1 - \mathcal{C}_\gamma)} \right)^{1/N_\gamma} - 1 \right] \tag{9}$$

where $k = \max\{k \in \mathbb{N}^0 | (k+1)\Pi - \Theta < T_{min}\}$. □

From Theorem 1, we can derive the following corollary.

Corollary 1. *Given a task $\tau = (T, C)$ and a single periodic resource $\gamma = (\Pi, \Theta)$ with condition $T \geq 2\Pi - \Theta$, the task τ is guaranteed to be schedulable on resource γ with the RM scheduling policy if*

$$\mathcal{C}_\gamma \cdot \frac{k}{k + 2(1 - \mathcal{C}_\gamma)} \geq U_\tau \tag{10}$$

where $k = \max\{k \in \mathbb{N}^0 | (k+1)\Pi - \Theta < T\}$. □

Proof. Task τ is guaranteed to be schedulable on resource γ under RM scheduling if

$$\mathtt{UB}_\gamma \geq U_\tau$$

Since there is only one task to be scheduled on resource γ, the utilization bound of γ can be obtained by substituting N_γ in formula (9) with 1. Hence, we have

$$\mathtt{UB}_\gamma = \mathcal{C}_\gamma \cdot \frac{k}{k + 2(1 - \mathcal{C}_\gamma)}$$

where $k = \max\{k \in \mathbb{N}^0 | (k+1)\Pi - \Theta < T\}$.

■

Definition 5. [Abstraction Overhead (O)][9] *For a single periodic resource γ and a task set Γ, the abstraction overhead is defined as*

$$O = \frac{\mathcal{C}_\gamma - U_\Gamma}{U_\Gamma} \tag{11}$$

Theorem 2. *[9] Given a single periodic resource γ and a task set Γ which is schedulable on γ under RM, the lower abstraction overhead bound is 0.443 when $k \to +\infty$, i.e.,*

$$\mathtt{OB} = \frac{\mathcal{C}_\gamma - U_\Gamma}{U_\Gamma} \geq 44.3\% \tag{12}$$

where $k = \max\{k \in \mathbb{N}^0 | (k+1)\Pi - \Theta < T_{min}\}$.

□

From Theorem 2, we can derive the following corollary:

Corollary 2. *Given a single periodic resource γ and a task set Γ which is schedulable on γ with RM, the upper bound of resource utilization rate of γ is 0.693 when $k \to +\infty$, i.e.,*

$$\mathtt{UR}_\gamma \leq 69.3\% \tag{13}$$

where $k = \max\{k \in \mathbb{N}^0 | (k+1)\Pi - \Theta < T_{min}\}$. □

Proof. According to Theorem 2, we have

$$\mathcal{C}_\gamma \geq 1.443 U_\Gamma \tag{14}$$

Since Γ is schedulable on γ with RM, by Definition 3, we have

$$\mathtt{UR}_\gamma = \frac{U_{\Gamma_\gamma}}{\mathcal{C}_\gamma} = \frac{U_\Gamma}{\mathcal{C}_\gamma} \leq 69.3\% \tag{15}$$

■

It is worth noting that Theorem 1 is only a sufficient condition with a strong precondition of $T_{\min} \geq 2\Pi - \Theta$. In other words, a task set may still be schedulable on a periodic resource even if the task set violates the utilization bound (Eq. (9)) or its precondition $T_{\min} \geq 2\Pi - \Theta$. For example, given a periodic resource $\gamma = (10, 5)$ and a periodic task $\tau = (10, 1)$. The task τ is schedulable on the resource γ, though $T < 2\Pi - \Theta$. However, for Theorem 2 and Corollary 2, the assumption made is that the task set Γ is schedulable on γ, which may or may not satisfy Eq. (9) or $T_{\min} \geq 2\Pi - \Theta$.

To achieve the above utilization rate of 69.3%, a precondition is that $k \to +\infty$, which implies that the period of the resource is infinitely small. When the resource period becomes infinitely small, the periodic resource approaches the dedicated resource. In this case, the utilization bound for a single periodic resource becomes the Liu and Layland RM utilization bound for dedicated resources [17]. However, since a resource with an infinitely small period is difficult to implement in reality, Shin's periodic resource utilization bound is hardly achievable in practice.

3.3 Problem Formulation

The problem to be addressed in this paper is how to assign a task set to a periodic resource set such that the used resource set utilization rate is maximized and all tasks are schedulable on their assigned resources. It is formally defined as follows:

Given a task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ where $\tau_i = (T_i, C_i)$, and a resource set $\mathcal{R} = \{\gamma_1, \gamma_2, \ldots, \gamma_M\}$ where $\gamma_j = (\Pi_j, \Theta_j)$, assign Γ to $\mathcal{R}$ such that:

Object: $\max \mathtt{UR}_\mathcal{R}$

Subject to:

Constraint 1: $\forall j\ 1 \leq j \leq M,\ \mathcal{C}_{\gamma_j} \cdot \frac{k}{k+2(1-\mathcal{C}_{\gamma_j})} \geq U_{\max}$

Constraint 2: $M \geq N$

Constraint 3: $\forall j\ 1 \leq j \leq M,\ 2\Pi_j - \Theta_j \leq T_{\min}$

where $k = \max\{k \in \mathbb{N}^0 | (k+1)\Pi_j - \Theta_j < T_{min}\}$.

Constraint 1 guarantees that each resource is large enough to schedule any single task in the given task set (Corollary 1). Constraint 2 together with Constraint 1 ensure that the task set is schedulable on the resource set without task splitting. Constraint 3 is the pre-condition of Constraint 1 (needed by Theorem 1).

Constraint 3 also implies $\Pi_j \leq T_{\min}$, i.e., all resource periods have to be smaller than or equal to task periods. This restriction is necessary for a task to be schedulable on a resource. Otherwise, even if the resource's capacity is larger than the task's utilization, the task may still not be schedulable on the resource. For example, given a periodic resource $\gamma = (20, 10)$ with capacity $\mathcal{C}_\gamma = 0.5$ and a periodic task $\tau = (10, 1)$ with utilization $U_\tau = 0.1$, if the given resource γ's available time occurrence is at the end of each period as shown in Fig. 1, the task τ is not schedulable on the resource γ even if $\mathcal{C}_\gamma > U_\tau$.

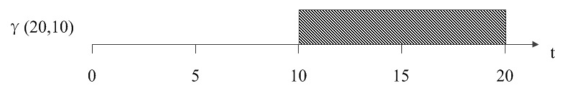


Fig. 1. Resource Occurrence Example

We take two steps to address the problem. First, we analyze the periodic resource harmonic utilization bound under the RM scheduling policy, and present task set harmonic transformation with respect to a periodic resource (Section 4). Second, we present the Best-Harmonically-Fit (BHF) algorithm (Section 5) which assigns tasks to resources with the goal of maximizing the resource set utilization rate while guaranteeing task set schedulability.

In a cloud computing environment, there are virtually unlimited resources to guarantee the execution of a given task set, however, how to optimally utilize these resources while guaranteeing real-time task deadlines is a research challenge. This paper is to address the challenge and presents an algorithm to maximize the resource utilization rate under the assumption that a given task set Γ is schedulable on the given resource set $\mathcal{R}$, i.e., three constraints defined in the formulated problem are satisfied.

4 Harmonic Property

4.1 Utilization Bound for Harmonically Related Task Set and Periodic Resource

If a harmonic task set is also harmonic with a given periodic resource, then the schedulable utilization bound of the task set can be as high as the resource capacity. To prove this property, we first give a lemma to show that if one task is harmonic with a given periodic resource, then the task can fully utilize the resource. We then prove a theorem stating that if a harmonic task set and a periodic resource are harmonic, the resource can be fully utilized by the task set under RM scheduling.

Lemma 1. *Given a task $\tau = (T, C)$ and a periodic resource $\gamma = (\Pi, \Theta)$, if the task and the resource are harmonic, i.e., $T = K \cdot \Pi$ $(K \in \mathbb{N}^+)$, then task τ is schedulable on γ if and only if $\frac{C}{T} \le \frac{\Theta}{\Pi}$.* □

Proof. Since $T = K \cdot \Pi$ $(K \in \mathbb{N}^+)$ and γ and τ all start at time 0, each task period T contains K complete resource periods. In other words, in each period, task τ obtains $K\cdot\Theta$ allocation time from resource γ. Hence, to guarantee that in each period of τ there is at least C allocation time, if and only if the following condition holds: $K \cdot \Theta \ge C$, which means $\frac{C}{T} \le \frac{K\cdot\Theta}{T}$, i.e. $\frac{C}{T} \le \frac{\Theta}{\Pi}$. ■

Lemma 2. *Given a task set Γ with two harmonic tasks $\tau_1 = (T, C_1)$ and $\tau_2 = (K \cdot T, C_2)$, and a periodic resource $\gamma = (\Pi, \Theta)$ which is also harmonic with the task set Γ. Let task $\tau = (T, C_1 + \frac{C_2}{K})$, if task τ is schedulable on resource γ, then the task set Γ is also schedulable on γ with RM scheduling.* □

Proof. Since τ is harmonic with γ and is schedulable, according to Lemma 1, we have $\frac{C_1+C_2/K}{T} \le \frac{\Theta}{\Pi}$.

In task set Γ, based on RM, τ_1 has the highest priority. In addition, since τ_1 and τ have the same period, both release at time 0, and $C_1 < C_1 + \frac{C_2}{K}$, if τ is schedulable, it guarantees that τ_1 is schedulable.

Since τ_2 also releases at time 0, within each period of τ_2, there are K instances of τ_1. The total resource demand is C_2+KC_1. Furthermore, within each period of τ_2, there are also K instances of τ, with total execution time of $K\times(C_1+\frac{C_2}{K}) = KC_1+C_2$. As τ is schedulable on resource γ, τ_2 is also schedulable on resource γ. ■

Based on Lemma 1 and Lemma 2, we have the following theorem.

Theorem 3. *Given a harmonic task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ with period $T_i = T_1 \cdot p^{i-1} (p \in \mathbb{N}^+)$ and a periodic resource $\gamma = (\Pi, \Theta)$ which is harmonic with the task set Γ, i.e., $T_1 = K \cdot \Pi (K \in \mathbb{N}^+)$, the harmonic utilization bound under RM scheduling policy is*

$$\mathtt{HUB}_\gamma = \frac{\Theta}{\Pi} \tag{16}$$

□

Proof. To prove Theorem 3 is equivalent to prove that the task set Γ is schedulable on the resource γ under RM scheduling if

$$\sum_{i=1}^{N} \frac{C_i}{T_i} \le \frac{\Theta}{\Pi} \tag{17}$$

holds.

We use induction on the number of tasks (n) in the task set to prove the theorem.

- Base case $n = 1$, based on Lemma 1, the theorem holds.
- Assume when $n = N$, if $\sum_{i=1}^{n} \frac{C_i}{T_i} \le \frac{\Theta}{\Pi}$, the task set is schedulable on γ with RM scheduling.
- We prove that if $n = N+1$ and $\sum_{i=1}^{N+1} \frac{C_i}{T_i} \le \frac{\Theta}{\Pi}$, then the task set is schedulable on the resource by RM. Without loss of generality, we assume the task set is $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_{N-1}, \tau_N, \tau_{N+1}\}$ and $T_i \le T_{i+1}, \forall i \in \{1, 2, ..., N\}$. Since Γ is harmonic, $\frac{T_{N+1}}{T_N} = p$ where $p \in \mathbb{N}^+$, we replace τ_N and τ_{N+1} by a single task $\tau = (T_N, C_N + \frac{C_{N+1}}{p})$ and denote the new task set as $\Gamma^* = \{\tau_1, \tau_2, ..., \tau_{N-1}, \tau\}$. The utilization of U_{Γ^*} is calculated as:

$$U_{\Gamma^*} = \sum_{i=1}^{N-1} \frac{C_i}{T_i} + \frac{C_N + \frac{C_{N+1}}{p}}{T_N} = \sum_{i=1}^{N+1} \frac{C_i}{T_i} = U_\Gamma \le \frac{\Theta}{\Pi}$$

  Hence, for Γ^*, its task number $n = N$ and its total utilization $U_{\Gamma^*} \le \frac{\Theta}{\Pi}$. Then, according to the induction assumption, Γ^* is schedulable, which indicates τ along with the task set $\{\tau_1, ..., \tau_{N-1}\}$ is schedulable by RM. According to Lemma 2, τ_N and τ_{N+1} are also schedulable if τ is schedulable. Hence, the task set Γ is schedulable by RM.

■

4.2 Task Set Harmonic Transformation with Respect to Periodic Resource

As discussed in the previous sub-section, the harmonic relation among a set of tasks and a resource brings the advantage that the resource capacity can be fully utilized by the task set. However, in a real world this criteria, i.e., tasks and resource are pairwise harmonic, is difficult to achieve. In order to take the advantage of the harmonic relationship, we need to transform an arbitrary task set into a task set that meets the criteria.

Han *et al.* [20] developed the DCT (Distance-Constrained Tasks) algorithm that transforms an arbitrary task set into a harmonic task set. In particular, given a task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$, if we use task τ_1's period T_1 as the base for transformation, the harmonic periods of the other tasks transformed by the DCT Algorithm are

$$T_i' = \begin{cases} T_1 \cdot \lfloor T_i/T_1 \rfloor & \text{if } T_i \geq T_1 \\ \frac{T_1}{\lceil T_1/T_i \rceil} & \text{if } T_i < T_1 \end{cases} \tag{18}$$

where $2 \leq i \leq N$.

Han *et al.* [20] also proved that such transformation does not change the schedulability of the given task set as shown in the following theorem.

Theorem 4. *[20] Given a task set $\Gamma = \{\tau_i(T_i, C_i)\}$, if there exists another task set $\Gamma' = \{\tau_i'(T_i', C_i')\}$ such that $T_i' \leq T_i$ and $C_i' = C_i$, for $1 \leq i \leq N$, and Γ' is schedulable by RM, then Γ is also schedulable by RM.* □

For the problem we are to address (defined in Section 3.3), the smallest task period in the task set and the resource must satisfy *Constraint 3*, i.e., $2\Pi - \Theta \leq T_{\min}$, which implies that $\Pi < T_{\min}$ holds. Based on (18) and *Constraint 3*, we give the definition of a task's harmonic transformation with respect to a resource.

Definition 6. *[Task Harmonic Transformation] Given a task $\tau = (T, C)$ and a periodic resource $\gamma = (\Pi, \Theta)$, let $\tau' = (T', C)$ where the period is*

$$T' = \Pi \cdot \lfloor T/\Pi \rfloor \tag{19}$$

Then τ' is called task τ's harmonic transformation with respect to resource γ. □

According to Theorem 3, tasks and the resource must be pairwise harmonic. When assigning more than one task to the same resource, the harmonic transformation must not only be harmonic with respect to the resource period, but also be harmonic with respect to the periods of the tasks that are already assigned to the resource. We use a recursive definition to define a task set's harmonic transformation with respect to a given resource.

Definition 7. *[Task Set Harmonic Transformation] Given a task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and a periodic resource $\gamma = (\Pi, \Theta)$,*

$$\Gamma' = \{\tau_i' = (T_i', C_i) \mid 1 \leq i \leq N\}$$

is task set Γ's harmonic transformation with respect to resource γ, where

$$T_1' = \Pi \cdot \lfloor T_1/\Pi \rfloor \tag{20}$$

$$T_i' = \max\{t \in \mathbb{N}^+ \mid 2 \leq i \leq N \tag{21}$$

$$\wedge\ t \leq T_i \wedge\ t \bmod \Pi = 0 \tag{22}$$

$$\wedge\ \forall j \in [1, i-1] : t \bmod T_j' = 0 \vee T_j' \bmod t = 0\} \tag{23}$$

□

In Definition 7, formula (20) guarantees that the first task is harmonic to the periodic resource. The formula (22) and (23) guarantee that all tasks are harmonic to the periodic resource and all prior tasks assigned to the resource, respectively. We use an example to illustrate the task set harmonic transformation.

Example 1. *Given a task set $\Gamma = \{\tau_1(13, 2), \tau_2(25, 4), \tau_3(20, 3)\}$, and a periodic resource $\gamma = (6, 4)$, we are to compute Γ's harmonic transformation with respect to γ based on Definition 7.*

The period of τ_1's harmonic transformation τ_1' is calculated directly by (20), so $\tau_1' = (12, 2)$.

The task τ_2's harmonic period with respect to γ is $T_2' = 24$, which is also harmonic with T_1'. Hence, the harmonic transformation of τ_2 is $\tau_2' = (24, 4)$.

If only considering the harmonic relationship between τ_3 and γ, then the harmonic period of τ_3 is $T_3' = 18$. However, $T_3' = 18$ is not harmonic with $T_1' = 12$ nor $T_2' = 24$. According to formula (23)*, τ_3's harmonic transformation becomes to $\tau_3' = (12, 3)$. Hence, the harmonic transformation of Γ with respect to γ is $\Gamma' = \{(12, 2), (24, 4), (12, 3)\}$. After the transformation, each task in Γ' is not only harmonic with the resource γ but also harmonic with all other tasks in Γ'.* □

The task set harmonic transformation defined in Definition 7 can be computed by Algorithm 1.

Algorithm 1 TRANSFORM(Γ, γ)

```
Input: A task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and a periodic resource $\gamma = (\Pi, \Theta)$.
Output: The task set $\Gamma$'s harmonic transformation $\Gamma' = \{\tau_1', \tau_2', \ldots, \tau_N'\}$ with respect to $\gamma$.
1:  $\Gamma' = \emptyset$, $\tau' =$ NULL
2:  $T_1' = \Pi \cdot \lfloor T_1/\Pi \rfloor$
3:  $\tau' = (T_1', C_1)$
4:  $\Gamma' = \Gamma' \cup \{\tau'\}$
5:  for $i = 2$ to $N$ do
6:    $T_i' = \Pi \cdot \lfloor T_i/\Pi \rfloor$
7:    while TRUE do
8:      if $\forall \tau_k' \in \Gamma' : T_k' \bmod T_i' = 0 \vee T_i' \bmod T_k' = 0$ then
9:        $\tau' = (T_i', C_i)$
10:       $\Gamma' = \Gamma' \cup \{\tau'\}$
11:       Break
12:     else
13:       $T_i' = T_i' - \Pi$
14:     end if
15:   end while
16: end for
17: return $\Gamma'$
```

In Algorithm 1, the first task's harmonic transformation is directly calculated by Eq. (20) (Line 2). For the remaining tasks, the `while` loop (Line 7 to Line 15) calculates the harmonic period T_i' with respect to the resource period Π (Line 6), and determines whether T_i' is also harmonic with all tasks already transformed, i.e., Γ' (Line 8 to Line 14). If T_i' is also harmonic with all tasks in Γ', the harmonic transformation of τ_i is found (Line 9 to Line 11); otherwise, T'_i is decreased by the resource period Π each time until T'_i is also harmonic with all tasks in Γ' (Line 13). The `while` loop is guaranteed to terminate, as in worst case when T'_i decreases to Π, it becomes harmonic to the resource and all tasks already transformed. The complexity of Algorithm 1 is $O(N^2)$.

The defined harmonic transformation does not change the task set's schedulability as shown in the following theorem.

Theorem 5. *Given a periodic resource $\gamma = (\Pi, \Theta)$, a task set Γ, and its harmonic transformation Γ' with respect to γ, if $U_{\Gamma'} \leq \Theta/\Pi$ the task set Γ is schedulable on resource γ under RM scheduling policy.* □

Proof. Definition 7 indicates $T'_i \leq T_i$ and $C'_i = C_i$, for $1 \leq i \leq N$. Based on Definition 7 and Theorem 4, under RM scheduling, if Γ' is schedulable, then Γ is also schedulable.

According to Theorem 3, $U_{\Gamma'} \leq \Theta/\Pi$ indicates that Γ' is schedulable on γ under RM scheduling. Hence, the task set Γ is schedulable on resource γ by RM. ■

To measure how harmonically related a task (set) and a resource (set) are, we introduce the following two definitions.

Definition 8. [Task and Resource Harmonicity $(H(\tau,\gamma))$] *Given a task $\tau = (T, C)$ and a periodic resource $\gamma = (\Pi, \Theta)$, assume the task's harmonic transformation with respect to the resource is $\tau' = (T', C)$. The harmonicity between task τ and resource γ is defined as*

$$H(\tau,\gamma) = \frac{T'}{T}. \tag{24}$$

□

Definition 9. [Task Set and Resource Set Harmonicity $(H(\Gamma, \mathcal{R}))$] *Given a task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and a periodic resource set $\mathcal{R} = \{\gamma_1, \gamma_2, \ldots, \gamma_M\}$, the harmonicity between task set Γ and resource set $\mathcal{R}$ is defined as the average value of each task's harmonicity with every resource, i.e.,*

$$H(\Gamma, \mathcal{R}) = \frac{\sum\limits_{\tau_i \in \Gamma, \gamma_j \in \mathcal{R}} H(\tau_i, \gamma_j)}{N \cdot M}. \tag{25}$$

□

Lemma 3. *The harmonicity $H(\tau,\gamma)$ of a task $\tau = (T, C)$ and a periodic resource $\gamma = (\Pi, \Theta)$ is in the range $(0.5, 1]$, i.e., $0.5 < H(\tau,\gamma) \leq 1$.* □

Proof. By the harmonic transformation definition (Definition 6) and harmonicity definition (Definition 8), we have $T' \leq T$, hence $H(\tau,\gamma) \leq 1$.

We prove $H(\tau,\gamma) > 0.5$ by contradiction. Since the period of a task is a positive number, the harmonicity must be larger than 0. Assume to the contrary, we have $0 < H(\tau,\gamma) \leq 0.5$. According to the harmonic transformation definition (Definition 6), we assume $T' = K \cdot \Pi$, where $K \in \mathbb{N}^+$. By the harmonicity definition (Definition 8), we have the following inequality

$$H(\tau,\gamma) = \frac{T'}{T} = \frac{K \cdot \Pi}{T} \leq 0.5 \tag{26}$$

hence

$$T \geq 2K \cdot \Pi \tag{27}$$

By formula (19) and formula (27), the period of τ' should be at least $2K \cdot \Pi$, i.e., $T' \geq 2K \cdot \Pi$, which contradicts the assumption that $T' = K \cdot \Pi$. Hence, the assumption $0 < H(\tau,\gamma) \leq 0.5$ does not hold, i.e., $H(\tau,\gamma) > 0.5$.

Therefore, we prove that $0.5 < H(\tau,\gamma) \leq 1$. ■

For a task τ, the utilization of its harmonic transformation is

$$U_{\tau'} = C/T' = U_\tau / H(\tau,\gamma) \tag{28}$$

and the utilization increment is

$$\Delta U_\tau = U_{\tau'} - U_\tau = (\frac{1}{H(\tau,\gamma)} - 1) \cdot U_\tau \tag{29}$$

which defines the utilization difference between a task and its harmonic transformation.

Lemma 4. *Given a task τ and a periodic resource γ, the utilization increment caused by task harmonic transformation is less than 100%, i.e.,*

$$0 \leq \frac{\Delta U_\tau}{U_\tau} < 1. \tag{30}$$

□

Proof. It can be directly derived from Lemma 3 and formula (29). ■

It is not difficult to see that there is a trade-off: when a task set is transformed to a harmonic task set with respect to the given periodic resource, the transformed task set can utilize the resource to its capacity with guaranteed schedulibility under RM. But on the other hand, the transformation itself increases the task set's utilization. Therefore, to reduce the cost of utilizing the harmonicity property, we need to select tasks and periodic resources that are best harmonically fit.

5 Best-Harmonically-Fit (BHF) Task Assignment Algorithm

As discussed in Section 4, the *harmonicity* measures how harmonically related a task (set) is to a resource (set). The higher the harmonicity, the smaller the task utilization increment ΔU_τ caused by the harmonic transformation. As the harmonic utilization bound of a periodic resource is fixed (Theorem 3), a resource has more potential availability to schedule other tasks if the task utilization increment ΔU_τ caused by the harmonic transformation is smaller, i.e., the harmonicity between the task and the resource is higher. In this section, we present the Best-Harmonically-Fit (BHF) task assignment algorithm that utilizes the harmonicity characteristics between tasks and periodic resources to maximize the periodic resource utilization rate. The intuition behind the proposed BHF task assignment algorithm is to assign tasks to a periodic resource in the non-decreasing order of their harmonicity to the resource until the resource can no longer schedule more tasks.

When assigning tasks to a resource, we have to ensure that all tasks assigned to the resource are schedulable, i.e., the total task utilization must not exceed the utilization bound. Shin's utilization bound (9) applies to a general task set, and is therefore relatively low. The harmonic utilization bound only applies to a harmonic task set that is also harmonic to the resources. Only when the harmonicity condition holds, can the task set fully utilize the resource's capacity. For an arbitrary task set, in order to take the advantage of the higher harmonic utilization bound, the task set has to be transformed to a harmonic task set. However, such transformation may result in an increased task utilization. Hence, to guarantee schedulability and also maximize the resource utilization rate, when deciding if a task set's utilization exceeds the resource's capacity, both Shin's utilization bound (9) and the harmonic utilization bound (16) need to be checked. The task set is schedulable on the resource if either bound is satisfied. Theorem 6 gives the combined schedulability condition.

Theorem 6. *Given a periodic resource γ, and a set of tasks Γ_γ assigned to the resource γ. For a new task τ, let τ' and Γ'_γ be the harmonic transformations of task τ and task set Γ_γ with respect to γ, respectively, the task τ is schedulable on γ if the following schedulability condition $SC(\tau,\gamma)$ is satisfied:*

$$SC(\tau,\gamma) : (U_{\Gamma'_\gamma} + U_{\tau'} \leq \texttt{HUB}_\gamma) \vee (U_{\Gamma_\gamma} + U_\tau \leq \texttt{UB}_\gamma) \quad (31)$$

□

Proof. The conclusion can be directly derived from Theorem 1 and Theorem 3. ■

We use an example to illustrate that both bounds are needed in deciding task schedulability on a given resource.

Example 2. *Given a periodic resource $\gamma = (7,5)$, assume there is no other task assigned to the resource. Decide if a task τ given below can be assigned to the resource.*
***Case** 1: $\tau = (12, 5.2)$*
The task's utilization $U_\tau = 5.2/12$ is smaller than Shin's utilization bound $\texttt{UB}_\gamma = 5/9$. Hence, the task can be assigned to the resource. However, task τ's harmonic transformation with respect to γ is $\tau' = (7, 5.2)$, with utilization $U_{\tau'} = 5.2/7$ which is larger than the resource's harmonic utilization bound, i.e., resource capacity $\texttt{HUB}_\gamma = 5/7$. In other words, task τ's harmonic transformation is not schedulable on the given resource. In this case, we need to use Shin's utilization bound (Theorem 1) to decide whether the task can be assigned to the resource.
***Case 2:** $\tau = (15, 9)$*
The task's utilization $U_\tau = 9/15$ is larger than Shin's utilization bound $\texttt{UB}_\gamma = 5/9$. Hence, the task shall not be assigned to the resource according to Shin's bound. However, task τ's harmonic transformation with respect to γ is $\tau' = (14, 9)$, with utilization $U_{\tau'} = 9/14$ which is smaller than the resource's harmonic utilization bound $\texttt{HUB}_\gamma = 5/7$. In other words, task τ's harmonic transformation is schedulable on the resource. Hence, τ is also schedulable on γ. In this case, we need to use the harmonic utilization bound (Theorem 3) to decide if τ can be assigned to γ.
***Case** 3: $\tau = (15, 5.2)$*
The task's utilization $U_\tau = 5.2/15$ is smaller than Shin's utilization bound $\texttt{UB}_\gamma = 5/9$. Hence, it is schedulable on the resource. Furthermore, task τ's harmonic transformation with respect to γ is $\tau' = (14, 5.2)$, with utilization $U_{\tau'} = 5.2/14$ which is also smaller than the resource's harmonic utilization bound $\texttt{HUB}_\gamma = 5/7$. In other words, task τ's harmonic transformation is also schedulable on the resource. In this case, we can use either Shin's utilization bound (Theorem 1) or the harmonic utilization bound (Theorem 3) to check schedulability. □

When a heuristic approach is used to assign tasks to periodic resources, the order in which tasks are assigned may impact the resource utilization rate. We use an example to illustrate this.

Example 3. *Given a task set $\Gamma = \{\tau_1(13,3), \tau_2(23,8), \tau_3(27,6), \tau_4(17,0.5)\}$ and a resource set $\mathcal{R} = \{\gamma_1(6,3), \gamma_2(5,2), \gamma_3(7,3.5)\}$, we assign the task set Γ to the resource set $\mathcal{R}$.*
If we first assign task τ_2 to resource γ_1, then the optimal way to assign the rest of the tasks is to assign τ_3 and τ_4 to γ_2, and τ_1 to γ_3. The resource utilization rate of such an assignment is 59.3%. However, if we first assign task τ_1 to resource γ_1, the optimal way to assign the rest of the tasks is to assign τ_3 to γ_1, and τ_2 and τ_4 to γ_3. Such assignment only uses two resources and increases the resource utilization rate to 83%. □

In this example, it is not difficult to see that τ_1 and γ_1 have the highest harmonicity among all task and resource pairs. The example also reveals that assigning the most harmonically related task and resource pair first leads to a higher resource utilization rate. Based on the observation and the discussion in Section 4 that assigning tasks to their most harmonically related resources can improve the resource utilization rate, we present the Best-Harmonically-Fit task set assignment algorithm. We first introduce the concept of Best-Harmonically-Fit-Task (BHFT) and Best-Harmonically-Fit-Pair (BHFP). Then, we show how to find the BHFT and the BHFP, respectively.

Definition 10. *Given a task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and a periodic resource γ, the Best-Harmonically-Fit-Task $\texttt{BHFT}(\Gamma,\gamma)$ with respect to resource γ is τ_i, i.e., $\texttt{BHFT}(\Gamma,\gamma) = \tau_i$, if task τ_i satisfies the following condition:*

$$\begin{aligned} &\tau_i \in \Gamma \wedge SC(\tau_i,\gamma) \\ &\qquad \wedge \forall j\ 1 \leq j \neq i \leq N\ H(\tau_i,\gamma) \geq H(\tau_j,\gamma) \\ &\qquad \wedge H(\tau_i,\gamma) = H(\tau_j,\gamma) \rightarrow U_{\tau_i} \geq U_{\tau_j} \end{aligned} \quad (32)$$

□

In other words, the $\texttt{BHFT}(\Gamma, \gamma)$ is the task in the task set Γ that satisfies: (1) it is schedulable on the resource, i.e., $SC(\tau_i, \gamma) = \texttt{true}$; (2) it has the highest harmonicity with resource γ, i.e., $\forall j\ 1 \le j \ne i \le N\ H(\tau_i, \gamma) \ge H(\tau_j, \gamma)$; (3) if more than one task in the task set has the same harmonicity with the given resource, it has the highest utilization, i.e., $\forall j\ 1 \le j \ne i \le N\ H(\tau_i, \gamma) = H(\tau_j, \gamma) \rightarrow U_{\tau_i} \ge U_{\tau_j}$.

We extend the best harmonically fit task with respect to a given resource to a given resource set and define the best harmonically fit task and resource pair.

Definition 11. *Given a task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and a resource set $\mathcal{R} = \{\gamma_1, \gamma_2, \ldots, \gamma_M\}$, the Best-Harmonically-Fit-Pair $\texttt{BHFP}(\Gamma, \mathcal{R})$ is (τ_i, γ_j), i.e., $\texttt{BHFP}(\Gamma, \mathcal{R}) = (\tau_i, \gamma_j)$, if the pair (τ_i, γ_j) satisfies the following condition:*

$$\begin{aligned} &\tau_i \in \Gamma, \gamma_j \in \mathcal{R} \\ &\wedge SC(\tau_i, \gamma_j) \\ &\wedge \forall m \forall k\ 1 \le m \ne i \le N 1 \le k \ne j \le M \\ &\quad H(\tau_i, \gamma_j) \ge H(\tau_m, \gamma_k) \\ &\wedge H(\tau_i, \gamma_j) = H(\tau_m, \gamma_k) \rightarrow U_{\tau_i} \ge U_{\tau_m} \end{aligned} \tag{33}$$

□

Algorithm 2 and Algorithm 3 give the pseudo code that find $\texttt{BHFT}(\Gamma, \gamma)$ and $\texttt{BHFP}(\Gamma, \mathcal{R})$, respectively. In Algorithm 2, the `for` loop (Line 4 to Line 10) compares all tasks in the task set against the given resource γ and finds the maximum $H(\tau_i, \gamma)$ and U_{τ_i}. The complexity of the algorithm is $O(N)$.

Algorithm 3 calls Algorithm 2 to find the BHFT for each resource in the resource set, and selects the task and resource pair that has the maximum harmonicity and task utilization. The complexity of Algorithm 3 is $O(NM)$.

Algorithm 2 SEARCH-BHFT(Γ, γ)

```
Input: A task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and a periodic resource $\gamma$.
Output: The BHFT $\tau_{\texttt{BHF}}$.
1: $\tau_{\texttt{BHF}} = \texttt{NULL}$
2: $H_{\max} = 0$
3: $U_{\max} = 0$
4: for $i = 1$ to $N$ do
5:     if $\texttt{SC}(\tau_i, \gamma) \wedge ((H(\tau_i, \gamma) > H_{\max}) \vee (H(\tau_i, \gamma) = H_{\max} \wedge U_{\tau_i} > U_{\max}))$ then
6:         $\tau_{\texttt{BHF}} = \tau_i$
7:         $H_{\max} = H(\tau_i, \gamma)$
8:         $U_{\max} = U_{\tau_i}$
9:     end if
10: end for
11: return $\tau_{\texttt{BHF}}$
```

Once the Best-Harmonically-Fit-Task and Best-Harmonically-Fit-Pair are found, we are ready to introduce the Best-Harmonically-Fit (BHF) task

Algorithm 3 SEARCH-BHFP$(\Gamma, \mathcal{R})$

```
Input: A task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and a periodic resource set $\mathcal{R} = \{\gamma_1, \gamma_2, \ldots, \gamma_M\}$.
Output: The BHFP $(\tau_{\texttt{BHF}}, \gamma_{\texttt{BHF}})$.
1: $\tau_{\texttt{BHF}} = \texttt{NULL}$
2: $\gamma_{\texttt{BHF}} = \texttt{NULL}$
3: $H_{\max} = 0$
4: $U_{\max} = 0$
5: for $i = 1$ to $M$ do
6:     $\tau_{tmp} = $ SEARCH-BHFT$(\Gamma, \gamma_i)$
7:     if $H(\tau_{tmp}, \gamma_i) > H_{\max} \vee (H(\tau_{tmp}, \gamma_i) = H_{\max} \wedge U_{\tau_{tmp}} > U_{\max})$ then
8:         $\tau_{\texttt{BHF}} = \tau_{tmp}$
9:         $\gamma_{\texttt{BHF}} = \gamma_i$
10:        $H_{\max} = H(\tau_{tmp}, \gamma_i)$
11:        $U_{\max} = U_{\tau_{tmp}}$
12:    end if
13: end for
14: return $(\tau_{\texttt{BHF}}, \gamma_{\texttt{BHF}})$
```

assignment algorithm given in Algorithm 4. In Algorithm 4, the `for` loop (Line 1 to Line 6) calculates the harmonic period T_i' of each task with respect to each resource, and the initial harmonicity value $H(\tau_i, \gamma_j)$ of every task and resource pair. If there are tasks in the task set Γ, Algorithm 3 is called to find the Best-Harmonically-Fit-Pair (Line 9). Once we have the best harmonically fit pair (τ, γ), assign the task τ to the resource γ (Line 11), remove τ from the given task set Γ (Line 12), compute the task τ's harmonic transformation τ' (Line 13), and once τ is assigned to the resource γ, we add the harmonic task τ' to the harmonic task set Γ_γ' which contains tasks assigned to the resource γ (Line 14). Update the harmonicity value for the remaining tasks with respect to γ (Line 15), and assign remaining tasks that are best harmonically fit to the resource γ until γ is full (Line 16 to Line 24) and then remove resource γ from the resource set $\mathcal{R}$ (Line 25). The complexity of Algorithm 4 is $O(N^2 M)$.

It is worth pointing out that in Algorithm 4, Line 13 and Line 20 update the harmonicity value between each unassigned task and the resource. According to Definition 7 and Definition 8, if a task is harmonically related to a resource, the task is also harmonically related to all the tasks assigned to the resource. Hence, we need to update tasks' harmonicity after each task assignment. The harmonicity update algorithm is shown in Algorithm 5.

The main steps of Algorithm 5 are similar to Algorithm 1. In Algorithm 5, each transformed task is harmonic to the harmonic transformation Γ_γ' of all tasks assigned to the resource γ, rather than all tasks already transformed in Algorithm 1. The `while` loop is guaranteed to terminate, as in worst case the transformed task period T_i' is the same as Π. The complexity of Algorithm 5 is $O(N^2)$.

Algorithm 4 BHF($\Gamma, \mathcal{R}$)

```
Input: A task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$ and a resource set
    $\mathcal{R} = \{\gamma_1, \gamma_2, \ldots, \gamma_M\}$.
    //Initialize task and resource harmonicity value
1: for $i = 1$ to $N$ do
2:     for $j = 1$ to $M$ do
3:         $T'_i = \Pi_j \cdot \lfloor T_i/\Pi_j \rfloor$
4:         $H(\tau_i, \gamma_j) = T'_i/T_i$
5:     end for
6: end for
7: $\tau =$ NULL, $\gamma =$ NULL
8: while $\Gamma \neq \emptyset$ do
9:     $(\tau, \gamma) =$SEARCH-BHFP($\Gamma, \mathcal{R}$)
       //Initialize the harmonic transformation of tasks
       assigned to $\gamma$
10:    $\Gamma'_\gamma = \emptyset$
11:    $\tau \mapsto \gamma$
       //Remove $\tau$ from $\Gamma$
12:    $\Gamma = \Gamma \setminus \{\tau\}$
       //Computer $\tau$'s harmonic transformation with re-
       spect to $\gamma$
13:    $\tau' = (T \cdot H(\tau, \gamma), C)$
       //Add $\tau'$ to $\Gamma'_\gamma$
14:    $\Gamma'_\gamma = \Gamma'_\gamma \cup \{\tau'\}$
       //Call Algorithm 5 to update remaining tasks'
       harmonicity with respect $\gamma$ and tasks already as-
       signed to $\gamma$
15:    UPDATE-H($\Gamma, \gamma, \Gamma'_\gamma$)
       //Call Algorithm 2 to find the BHFT
16:    $\tau =$SEARCH-BHFT($\Gamma, \gamma$)
17:    while $\tau! =$ NULL do
18:        $\tau \mapsto \gamma$
19:        $\Gamma = \Gamma \setminus \{\tau\}$
20:        $\tau' = (T \cdot H(\tau, \gamma), C)$
21:        $\Gamma'_\gamma = \Gamma'_\gamma \cup \{\tau'\}$
22:        UPDATE-H($\Gamma, \gamma, \Gamma'_\gamma$)
23:        $\tau =$SEARCH-BHFT($\Gamma, \gamma$)
24:    end while
25:    $\mathcal{R} = \mathcal{R} \setminus \{\gamma\}$
26: end while
```

Algorithm 5 UPDATE-H($\Gamma, \gamma, \Gamma'_\gamma$)

```
Input: A task set $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_N\}$, a periodic re-
    source $\gamma = (\Pi, \Theta)$, and the harmonic transformation
    $\Gamma'_\gamma$ of all tasks assigned to the resource $\gamma$.
1: for $i = 1$ to $N$ do
2:     $T'_i = \Pi \cdot \lfloor T_i/\Pi \rfloor$
3:     while TRUE do
4:         if $\forall \tau'_k \in \Gamma'_\gamma : T'_k \bmod T'_i = 0 \vee T'_i \bmod T'_k = 0$
           then
5:             $H(\tau_i, \gamma) = T'_i/T_i$
6:             Break
7:         else
8:             $T'_i = T'_i - \Pi$
9:         end if
10:    end while
11: end for
```

We use an example to illustrate the BHF procedure.

Example 4. *Consider the same resource set and task set as given in Example 3, i.e.,* $\Gamma = \{\tau_1(13,3), \tau_2(23,8), \tau_3(27,6), \tau_4(17,0.5)\}$, $\mathcal{R} = \{\gamma_1(6,3), \gamma_2(5,2), \gamma_3(7,3.5)\}$. *We are to use the proposed BHF algorithm to assign the task set to the resource set.*

We first calculate the harmonicity for each task and resource pair. The results are shown in Table 1(a).

The Best-Harmonically-Fit-Pair in Table 1(a) is (τ_1, γ_1). *Hence,* τ_1 *is assigned to* γ_1. *After* τ_1 *is assigned to* γ_1, *the harmonicity between each unassigned task and* γ_1 *is updated and shown in Table 1(b). From Table 1(a) and Table 1(b), it can be seen that the harmonicity of* τ_2 *decreases from* $18/23$ *to* $12/23$.

Then, we find the Best-Harmonically-Fit-Task for resource γ_1, *which is* τ_3. *Task* τ_3 *is assigned to resource* γ_1.

Once τ_3 *is assigned to* γ_1, γ_1 *does not have enough capacity to host neither* τ_2 *nor* τ_4. *Hence, we find the next Best-Harmonically-Fit-Pair which is* (τ_2, γ_3), *and assign* τ_2 *to* γ_3. *The harmonicity of the only remaining task* τ_4 *with respect to* γ_3 *is updated to* $7/17$ *after assigning* τ_2 *to* γ_3.

Finally, we assign τ_4 *to resource* γ_3. *The harmonicity between each task and each resource is shown in Table 1(c).*

The task assignment resulted from the BHF algorithm for this example matches the optimal assignment given in Example 3, with a resource utilization rate of 83%. □

TABLE 1
Harmonicity for Γ and $\mathcal{R}$

(a) Initial State (Before Assignment)

$H(\tau_i, \gamma_j)$	γ_1	γ_2	γ_3
τ_1	12/13	10/13	7/13
τ_2	18/23	20/23	21/23
τ_3	24/27	25/27	21/27
τ_4	12/17	15/17	14/17

(b) After Assigning τ_1 to γ_1

$H(\tau_i, \gamma_j)$	γ_1	γ_2	γ_3
τ_1	12/13	10/13	7/13
τ_2	**12/23**	20/23	21/23
τ_3	24/27	25/27	21/27
τ_4	12/17	15/17	14/17

(c) Final State (Assignment Done)

$H(\tau_i, \gamma_j)$	γ_1	γ_2	γ_3
τ_1	12/13	10/13	7/13
τ_2	**12/23**	20/23	21/23
τ_3	24/27	25/27	21/27
τ_4	12/17	15/17	**7/17**

The BHF algorithm is a heuristic and uses a sufficient utilization bound to check schedulability. The scenario where a task set violates the bound but is still schedulable exists and the necessary utilization bound for RM scheduling on periodic resources is yet to be found.

In next section, we experimentally evaluate how well the BHF algorithm performs when it is compared with other heuristic approaches and the optimal solutions found through exhaustive search.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2015.2437379, IEEE Transactions on Parallel and Distributed Systems

6 Performance Evaluation

In this section, we experimentally evaluate the performance of the proposed Best-Harmonically-Fit algorithm through simulations. We compare the BHF task assignment algorithm with three commonly used multiprocessor task assignment algorithms, namely Best-Fit Decreasing (BFD), First-Fit Decreasing (FFD), and Worst-Fit Decreasing (WFD) algorithms [16], which assign tasks in non-increasing utilization order.

When BFD, FFD, and WFD are used, Shin's utilization bound [9], i.e., formula (9), is used to decide if a task set assigned to a resource is schedulable.

- Best-Fit Decreasing (BFD) [16]: Assign task $\tau \in \Gamma$ to the periodic resource $\gamma \in \mathcal{R}$ so that the remaining capacity percentage is minimal after the assignment, i.e.,

$$U_{\Gamma_\gamma} + U_\tau \leq \mathtt{UB}_\gamma \quad \wedge$$
$$\frac{\mathtt{UB}_\gamma - U_{\Gamma_\gamma} - U_\tau}{U_\gamma} = \min\{\frac{\mathtt{UB}_{\gamma_j} - U_{\Gamma_{\gamma_j}} - U_\tau}{U_{\gamma_j}} | \forall \gamma_j \in \mathcal{R}\}$$

- First-Fit Decreasing (FFD) [16]: Assign task $\tau \in \Gamma$ to the first periodic resource $\gamma \in \mathcal{R}$ that satisfies τ's schedulability condition, i.e.,

$$U_{\Gamma_\gamma} + U_\tau \leq \mathtt{UB}_\gamma$$

- Worst-Fit Decreasing (WFD) [16]: Assign task $\tau \in \Gamma$ to the periodic resource $\gamma \in \mathcal{R}$ so that the remaining capacity percentage is maximal after the assignment, i.e.,

$$U_{\Gamma_\gamma} + U_\tau \leq \mathtt{UB}_\gamma \quad \wedge$$
$$\frac{\mathtt{UB}_\gamma - U_{\Gamma_\gamma} - U_\tau}{U_\gamma} = \max\{\frac{\mathtt{UB}_{\gamma_j} - U_{\Gamma_{\gamma_j}} - U_\tau}{U_{\gamma_j}} | \forall \gamma_j \in \mathcal{R}\}$$

The performance of a task assignment algorithm is evaluated by two criteria, i.e., (1) resource utilization rate $\mathtt{UR}_\mathcal{R}$, and (2) total number of periodic resources used $M_{\mathtt{used}}$. The higher the $\mathtt{UR}_\mathcal{R}$ and the smaller the $M_{\mathtt{used}}$, the better the performance of the algorithm.

In the following experiments, the task sets and the resource sets are generated using the UUniFast algorithm [27] which gives an unbiased distribution of utilization values.

6.1 Harmonicity Impact

This set of experiments is to evaluate harmonicity impact on the performance of task assignment algorithms.

Experiment Settings

- Task number: 20
- Task utilization range: $[0.1, 1.0]$
- Resource number: 20
- Resource capacity range: $[0.5, 1.0]$
- Resource set capacity: 13
- Harmonicity: varies in the range of $[0.55, 1.0]$ with step 0.05

As the harmonicity can never be smaller than 0.5 which is proven in Lemma 3, the experiment only considers harmonicities larger than 0.5.

Experiments

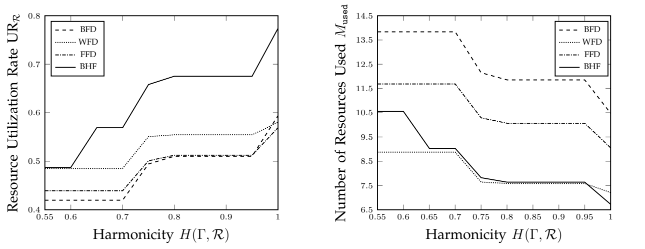
In this set of experiments, we randomly generate 200 resource sets; for each resource set, we randomly generate 10 task sets that satisfy *Constraint 1, Constraint 2,* and *Constraint 3* given in Section 3.3. For each valid task set, we fix the utilization of each task and adjust task periods to generate 10 task sets such that the harmonicity between the resource set and each of the task set is one of the values in harmonicity variable set $\{0.55, 0.6, 0.65, \ldots, 1.00\}$. For each test case, we apply the BFD, the FFD, the WFD, and the BHF algorithms to assign the generated tasks to resources. The average value of $200 \times 10 \times 10$ repeats is used to represent the performance of each algorithm.

Fig. 2(a) and Fig. 2(b) show the resource utilization rate and the number of periodic resources used under different harmonicities, respectively. From Fig. 2, we have the following observations:

1) For all four task assignment algorithms, the resource utilization rate increases and the number of periodic resources used decreases when harmonicity increases;
2) The BHF algorithm has up to 35.96% higher resource utilization rate and uses up to 55.79% less number of resources than the other three algorithms;
3) The BHF algorithm is more sensitive to the harmonicity change than the other three algorithms.

The third observation is consistent with the design of the BHF algorithm, i.e., BHF is based on harmonicity. The sensitivity of BHF is demonstrated in the following two aspects:

1) The discrepancy of the resource utilization rate between BHF and the other three algorithms increases with an increase in harmonicity;
2) The BHF has a larger increase of the resource utilization rate than the other three algorithms as the harmonicity increases.



(a) Utilization Rate under different Harmonicity (b) Number of Resources Used under different Harmonicity

Fig. 2. Harmonicity Impact

6.2 Task Set Utilization Impact

The second set of experiments evaluates the performance of the proposed BHF task assignment algorithm under different task set utilizations. The experiment parameters are given below.

Experiment Settings

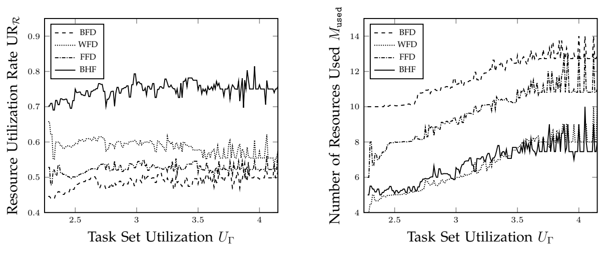
- Task number: 20
- Task utilization range: $[0.1, 1.0]$
- Task set utilization: random
- Resource number: 20
- Resource capacity range: $[0.3, 1.0]$
- Resource set capacity: 13

Experiments

In the experiments, we randomly generate 200 resource sets with above resource set parameters. For each resource set, we randomly generate 100 task sets that satisfy the three constraints given in Section 3.3. For each test case, we apply the four different task assignment algorithms. We run 200×100 test cases, and combine the results based on task set utilizations rounded to the nearest hundredth. The average value is used to represent the performance of each algorithm.

Fig. 3(a) depicts the average resource utilization rate under different task set utilizations. Among the four task assignment algorithms, the BHF algorithm has the highest resource utilization rate. The resource utilization rate resulted from the BHF algorithm is always above 69.05%. On average, the BHF algorithm results in a 53.26%, 42.54%, and 27.79% higher resource utilization rate than the BFD, the FFD, and the WFD algorithms, respectively.

Fig. 3(b) depicts the number of periodic resources used under different task set utilizations. In general, when the task set utilization increases, the number of resources used also increases. From Fig. 3(b), we observe that BFD uses the most number of resources, which is consistent with the observation that BFD has the lowest resource utilization rate. The WFD, on the other hand, uses less number of resources than the proposed BHF approach, but also has a lower utilization rate, which seems to be counter intuitive.

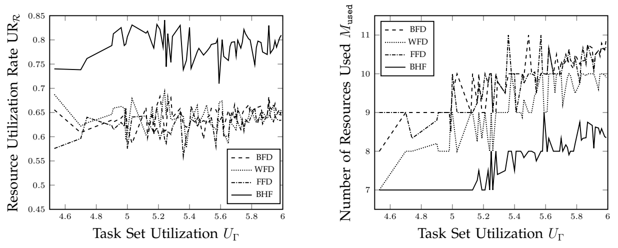


(a) Utilization Rate (b) Number of Resources Used

Fig. 3. Task Set Utilization Impact ($\frac{\Theta}{\Pi} \in [0.325, 1.0]$)

Further study reveals that since the WFD algorithm always selects the largest resource to assign tasks to, when resource capacity is relatively small, the remaining portion of the resource resulting from the WFD algorithm would have a higher possibility than other approaches to host another task. Therefore, the WFD algorithm results in less number of resources used. To verify this reasoning, we repeat the above experiments but with a higher capacity resource set: each resource capacity is in the range of $[0.8, 1.0]$ and we do not fix the total resource set capacity. The results are shown in Fig. 4.

As shown in Fig. 4, BHF has better performance (higher resource utilization rate and smaller number of resources used) than the other three algorithms. In particular, the BHF algorithm uses 23.96% less number of resources than the other three approaches on average. Another observation from Fig. 4 is that the performance difference among BFD, FFD, and WFD algorithms is small when individual resource capacity is large. The experiment also confirms our observation that when individual resource capacity is small, the WFD algorithm may have some advantage with respect to the number of resources needed.



(a) Utilization Rate (b) Number of Resources Used

Fig. 4. Task Set Utilization Impact ($\frac{\Theta}{\Pi} \in [0.8, 1.0]$)

6.3 BHF and Optimal Solution Comparison

The third set of experiments compares the performance of the proposed BHF task assignment algorithm with the optimal solution obtained by brute-force search. The experiment parameters are given below.

Experiment Settings

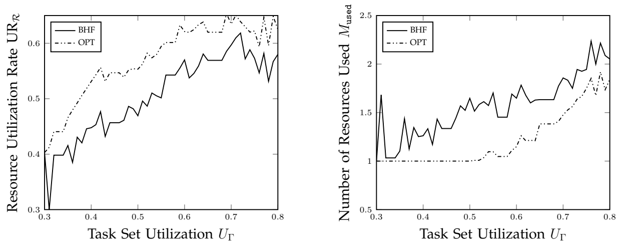
- Task number: 3
- Task utilization range: $[0.1, 1.0]$
- Task set utilization: random
- Resource number: 3
- Resource capacity range: $[0.3, 1.0]$
- Resource set capacity: 1.95

Experiments

The experiment procedure is the same as in Section 6.2 except for some parameters.

For each test case, we apply the four different task assignment algorithms, and consider all possible task assignments. We choose the one with the largest resource utilization rate as the optimal solution. It is worth noting that when searching for the optimal solution, we use both Shin's utilization bound (Theorem 1) and the harmonic utilization bound (Theorem 3) to check for schedulability. We consider tasks to be schedulable if either bound is satisfied.

Fig. 5(a) and Fig. 5(b) depict the average resource utilization rate and the number of resources used, respectively. Compared with the optimal solution, on average, the BHF algorithm results in 11.63% lower resource utilization rate and uses 22.02% more number of resources.



(a) Utilization Rate (b) Number of Resources Used

Fig. 5. BHF and Optimal Solution Comparison

7 Conclusion

Periodic resource models and their scheduling problems have drawn more attention in real-time community in recent years. However, to our best knowledge, there has not been much work, if any, in the literature dealing with the task assignment problem on multiple periodic resources. In this paper, we study the task assignment problem in the context of assigning multiple periodic tasks to multiple periodic resources. Specifically, we first study the harmonic properties between periodic tasks and periodic resources. We prove that if a harmonic task set is also harmonic with the resource, the task set can 100% utilize the resource's capacity under the RM scheduling algorithm. Then we propose a heuristic Best-Harmonically-Fit (BHF) task assignment algorithm to maximize the resource utilization rate based on the harmonic properties between periodic tasks and periodic resources. We compare the performance of Best-Harmonically-Fit (BHF) task assignment algorithm with Best-Fit Decreasing (BFD), First-Fit Decreasing (FFD), Worst-Fit Decreasing (WFD) task assignment algorithms, and the optimal task assignment (found through exhaustive search for a small-sized task set and resource set). The experiment concludes that, on average, the BHF algorithm results in 53.26%, 42.54%, and 27.79% higher resource utilization rate than the Best-Fit Decreasing(BFD), the First-Fit Decreasing (FFD), and the Worst-Fit Decreasing (WFD) task assignment algorithms, respectively; but comparing to the optimal resource utilization rate found by exhaustive search, it is about 11.63% lower.

Currently, we are developing a private cloud CODE-Cloud (CCloud) [28] on top of RT-Xen hypervisor, which extends Xen [29] with built-in real-time schedulers for periodic resources [30], [31], [32]. Our future work is to integrate the proposed BHF task assignment algorithm into RT-Xen scheduler.

Acknowledgement

The research is supported in part by NSF under grant number CAREER 0746643, CNS 1018731 and CNS 1035894.

References

[1] Aloysius K Mok, Xiang Feng, and Deji Chen. Resource partition for real-time systems. In *Real-Time Technology and Applications Symposium, 2001. Proceedings. Seventh IEEE*, pages 75–84. IEEE, 2001.

[2] Yu Li, Albert MK Cheng, and Aloysius K Mok. Regularity-based partitioning of uniform resources in real-time systems. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2012 IEEE 18th International Conference on*, pages 368–377. IEEE, 2012.

[3] Jaewoo Lee, Sisu Xi, Sanjian Chen, Linh TX Phan, Chris Gill, Insup Lee, Chenyang Lu, and Oleg Sokolsky. Realizing compositional scheduling through virtualization. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*, pages 13–22. IEEE, 2012.

[4] S Shirero, Matsumoto Takashi, and Hiraki Kei. On the schedulability conditions on partial time slots. In *Real-Time Computing Systems and Applications, 1999. RTCSA'99. Sixth International Conference on*, pages 166–173. IEEE, 1999.

[5] Aloysius K Mok and Xiang Alex. Towards compositionality in real-time resource partitioning based on regularity bounds. In *Real-Time Systems Symposium, 2001.(RTSS 2001). Proceedings. 22nd IEEE*, pages 129–138. IEEE, 2001.

[6] Insik Shin and Insup Lee. Periodic resource model for compositional real-time guarantees. In *Real-Time Systems Symposium, 2003. RTSS 2003. 24th IEEE*, pages 2–13, Dec 2003.

[7] Arvind Easwaran, Insik Shin, Oleg Sokolsky, and Insup Lee. Incremental schedulability analysis of hierarchical real-time components. In *Proceedings of the 6th ACM &Amp; IEEE International Conference on Embedded Software*, EMSOFT '06, pages 272–281, New York, NY, USA, 2006. ACM.

[8] A. Easwaran, Insup Lee, Insik Shin, and O. Sokolsky. Compositional schedulability analysis of hierarchical real-time systems. In *Object and Component-Oriented Real-Time Distributed Computing, 2007. ISORC '07. 10th IEEE International Symposium on*, pages 274–281, May 2007.

[9] Insik Shin and Insup Lee. Compositional real-time scheduling framework with periodic model. *ACM Transactions on Embedded Computing Systems (TECS)*, 7(3):30:1–30:39, May 2008.

[10] Nathan Fisher and Farhana Dewan. Approximate bandwidth allocation for compositional real-time systems. In *Real-Time Systems, 2009. ECRTS'09. 21st Euromicro Conference on*, pages 87–96. IEEE, 2009.

[11] Farhana Dewan and Nathan Fisher. Approximate bandwidth allocation for fixed-priority-scheduled periodic resources. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, pages 247–256. IEEE, 2010.

[12] Xiayu Hua, Zheng Li, Hao Wu, and Shangping Ren. Scheduling periodic tasks on multiple periodic resources. In *International Conference on Advanced Communications and Computation, 2014. INFOCOMP 2014. 4th IARIA*. IARIA, 2014.

[13] Yingfeng Oh and Sang H. Son. Tight performance bounds of heuristics for a real-time scheduling problem. Technical report, Charlottesville, VA, USA, 1993.

[14] Sudarshan K. Dhall and C. L. Liu. On a real-time scheduling problem. *Operations Research*, 26(1):127–140, 1978.

[15] Nan Guan, M. Stigge, Wang Yi, and Ge Yu. Fixed-priority multiprocessor scheduling with liu and layland's utilization bound. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, pages 165–174, April 2010.

[16] Sanjoy Baruah. Partitioned edf scheduling: a closer look. *Real-Time Systems*, 49(6):715–729, 2013.

[17] C. L. Liu and James W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM*, 20(1):46–61, jan 1973.

[18] A. Burchard, J. Liebeherr, Yingfeng Oh, and S.H. Son. New strategies for assigning real-time tasks to multiprocessor systems. *Computers, IEEE Transactions on*, 44(12):1429–1442, Dec 1995.

[19] Sylvain Lauzac, Rami Melhem, and Daniel Mossé. An efficient rms admission control and its application to multiprocessor scheduling. In *Parallel Processing Symposium(IPPS/SPDP), 1998*, pages 511–518. IEEE, 1998.

[20] Ching-Chih Han and Hung 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*, pages 36–45, Dec 1997.

[21] Björn Andersson, Sanjoy Baruah, and Jan Jonsson. Static-priority scheduling on multiprocessors. In *Real-Time Systems Symposium, 2001.(RTSS 2001). Proceedings. 22nd IEEE*, pages 193–202. IEEE, 2001.

[22] Ming Fan and Gang Quan. Harmonic-aware multi-core scheduling for fixed-priority real-time systems. *Parallel and Distributed Systems, IEEE Transactions on*, PP(99):1–11, 2013.

[23] Jan Korst, Emile Aarts, JanKarel Lenstra, and Jaap Wessels. Periodic multiprocessor scheduling. In Emile H.L. Aarts, Jan van Leeuwen, and Martin Rem, editors, *PARLE '91 Parallel Architectures and Languages Europe*, volume 505 of *Lecture Notes in Computer Science*, pages 166–178. Springer Berlin Heidelberg, 1991.

[24] Ming Fan and Gang Quan. Harmonic-fit partitioned scheduling for fixed-priority real-time tasks on the multiprocessor platform. In *Embedded and Ubiquitous Computing (EUC), 2011 IFIP 9th International Conference on*, pages 27–32, Oct 2011.

[25] Ming Fan and Gang 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*, pages 503–508, March 2012.

[26] Xiang Feng and Aloysius K Mok. A model of hierarchical real-time virtual resources. In *Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE*, pages 26–35. IEEE, 2002.

[27] Enrico Bini and GiorgioC. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154, 2005.

[28] Codecloud (ccloud). http://code.cs.iit.edu:8080/.

[29] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the art of virtualization. In *Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles*, SOSP '03, pages 164–177, New York, NY, USA, 2003. ACM.

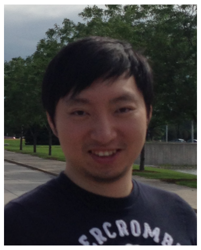
[30] Sisu Xi, J. Wilson, Chenyang Lu, and C. Gill. Rt-xen: Towards real-time hypervisor scheduling in xen. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*, pages 39–48, Oct 2011.

[31] Jaewoo Lee, Sisu Xi, Sanjian Chen, L.T.X. Phan, C. Gill, Insup Lee, Chenyang Lu, and O. Sokolsky. Realizing compositional scheduling through virtualization. In *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*, pages 13–22, April 2012.

[32] Sisu Xi, Meng Xu, Chenyang Lu, Linh T. X. Phan, Christopher Gill, Oleg Sokolsky, and Insup Lee. Real-time multi-core virtual machine scheduling in xen. In *Proceedings of the 14th International Conference on Embedded Software*, EMSOFT '14, pages 27:1–27:10, New York, NY, USA, 2014. ACM.

Chunhui Guo is now a Ph.D candidate in the Computer Science Department at Illinois Institute of Technology. He earned his BSEE and MSEE from Shandong University, China, in 2010 and 2013, respectively. His current research interests mainly focus on real-time systems and Cyber-Physical System.

Xiayu Hua is now a Ph.D candidate in the Computer Science Department at Illinois Institute of Technology. His research interests are in distributed file system, virtualization technology, real-time scheduling and cloud computing. He earned his B.S. degree from the Northwestern Polytechnic University, China, in 2008 and his M.S. degree from the East China Normal University, China, in 2012.



Hao Wu is now a Ph.D candidate in the Computer Science Department at Illinois Institute of Technology. He received B.E. in Information Security from Sichuan University, Chengdu, China, 2007. He received M.S. in Computer Science from University of Bridgeport, Bridgeport, CT, 2009. His current research interests mainly focus on cloud computing, real-time distributed open systems, Cyber-Physical System, parallel and distributed systems, and real-time applications.

Douglas Lautner is now a Ph.D candidate in the Computer Science Department at Illinois Institute of Technology. His research interests are in real-time wireless sensing and Cyber-Physical Systems. He earned his MSCS and MSEE from the Illinois Institute of Technology and holds an MBA from Tulane University. Doug is also a Director of Software at Motorola. He manages the WISL (Wireless, Internet of Things, Sensors and Location) Department. Doug currently has 15 patent pursues.

Dr. Shangping Ren is an associate professor in the Computer Science Department at the Illinois Institute of Technology. She earned her Ph.D from UIUC in 1997. Before she joined IIT in 2003, she worked in software and telecommunication companies as a software engineer and then lead software engineer. Her current research interests include coordination models for real-time distributed open systems, real-time, fault-tolerant and adaptive systems, Cyber-Physical System, parallel and distributed systems, cloud computing, and application-aware many-core virtualization for embedded and real-time applications.