Received May 14, 2019, accepted July 26, 2019, date of publication August 2, 2019, date of current version August 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932799

Co-Scheduling of Hybrid Transactions on Multiprocessor Real-Time Database Systems

CHENGGANG DENG[1], GUOHUI LI[2], QUAN ZHOU[1], AND JIANJUN LI[1]
[1]Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[2]Department of Software Engineering, Huazhong University of Science and Technology, Wuhan 430074, China

Corresponding author: Quan Zhou (quanzhou@hust.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61672252 and Grant 61802137, in part by the Fundamental Research Funds for the Central Universities under Grant 2019kfyXKJC021, and in part by the Hubei Provincial Natural Science Foundation of China under Grant 2018CFB204.

ABSTRACT Deriving deadlines and periods for update transactions and scheduling hybrid transactions have been recognized as an important issue in real-time database systems. Despite years of research, all existing techniques focus on assigning deadlines and periods for update transactions in uniprocessor systems and, thus, cannot be applied in multiprocessor systems. Two partitioned scheduling methods for hybrid transactions in multiprocessor systems are proposed in this paper, by developing existing scheduling methods for uniprocessor systems to multiprocessor environments. An effective strategy is proposed to improve the efficiencies of the proposed methods, under which the improvement of efficiency is at the price of the acceptance ratio of transaction sets. For addressing the low acceptance ratio of the improvement strategy, we propose a novel scheduling method. A set of experiments is conducted to test the performances of the proposed methods. To the best of our knowledge, this is the first paper for co-scheduling of hybrid transactions on multiprocessor platforms.

INDEX TERMS Real-time database systems, multiprocessor, hybrid transactions, deadlines and periods, co-scheduling.

I. INTRODUCTION

Real-time database systems (RTDBS) have been widely used in many applications that have strict requirements on data freshness. Typical applications of real-time database systems include industrial control [1], vehicular control [2], aerospace control [3], health monitoring [4] and robot control [5]. The main goal of these systems is to guarantee a certain performance in monitoring the external environment for generating timely and appropriate responses to critical events. Thus, the temporal validity of data objects is the key characteristic of these systems, which means the sampled values of data objects are valid only in some certain time intervals and systems need to update the values of data objects in their corresponding valid intervals.

To satisfy the key characteristic of real-time sensing and control systems, there are two types of transactions in RTDBSs: control transactions and update transactions. Control transactions access and use the data in systems and update transactions are responsible for monitoring real-time data objects and updating their values. Each control transaction is usually periodic and have a hard or firm deadline on each of its invocation. Scheduling control transactions to meet their deadlines is a typical real-time scheduling problem, and existing scheduling algorithms, including Earliest Deadline First (EDF) [6] and Deadline Monotonic (DM) [7], can be utilized. But it is worth noting that only satisfying the deadline constraints of control transactions is not enough to perform the monitoring and reaction functions effectively, since control transactions generally need to use *real-time data objects* maintained in RTDBSs to conduct appropriate reactions [8]. Therefore, another important problem in designing RTDBSs is to maintain the freshness, i.e., temporal validity (also called temporal consistency) [9], of real-time data objects. Each update transaction maintains only one data object and each data object is sampled at the release time of the update

The associate editor coordinating the review of this manuscript and approving it for publication was Mario Collotta.

transaction maintaining it. So, for each data object sampled at a time unit t, the update transaction maintaining it should be completed at least twice in $[t, t+x)$, where x is the valid interval length of the sample (at t) of this data object. The first completion of the update transaction is used to write the sampled value (at t) into RTDBSs and the second completion of the update transaction is used to update the value of this data object before the sampled value (at t) expires. Deriving deadlines and periods for update transactions and scheduling hybrid transactions receives much research attention in the field of RTDBSs.

Various deadline and period deriving and transaction scheduling methods are proposed in the past decades. But, most of them focus on uniprocessor environments [8], [10]–[12]. Different from the scheduling in uniprocessor systems, that for hybrid transaction set in multiprocessor systems needs to consider the assignment of transactions on processors. A good scheduling method for multiprocess systems should have high efficiency, high acceptance ratio and low total workload of processors.

In this paper, we consider the problem of co-scheduling of hybrid transactions on multiprocessor systems. The contributions of this paper are summarized as follows:

- Two intuitive partitioned scheduling methods (for hybrid transaction sets in multiprocessor systems) are proposed by developing some existing scheduling methods to multiprocess environment.
- For improving the efficiency of these two methods, we propose two improvement strategies which reduce the time for testing the schedulability of transaction sets. We also analyze the proposed improvement strategies and prove that their theoretical resource augmentations are equal to $3 - 1/m$, where m is the number of processors.
- A set of experiments is conducted to evaluate the performances of the proposed methods.

The rest of this paper is organized as follows: Section II discusses the related work. Section III reviews the definition of temporal validity and gives the notations used in this paper, as well as defines the problem studied in this work. Section IV gives two intuitive scheduling methods (for hybrid transactions in multiprocessor systems) by developing some existing scheduling methods for uniprocessor systems to multiprocessor environments. Section V proposes two effective scheduling methods to improve the efficiency of the proposed intuitive scheduling methods. Section VI shows the performances of our proposed methods, followed by a conclusion in Section VII.

II. RELATED WORK

There has been a lot of work for maintaining real-time data freshness in real-time data-intensive applications [13]–[17]. Song and Liu [18] studied the performances of the two-phase locking and the optimistic algorithm in maintaining temporal consistency of shared data in hard real-time systems with periodic tasks. Kuo and Mok [19] investigated real-time data-semantics and proposed the SSP protocol. Ho *et al.* [10] proposed a semantics-based reconfiguration method by combining the Half-Half (HH) scheme with similarity-based principles. The proposed method realizes the balance between data precision and processor workload. Gustafsson and Hansson [2] proposed the ODTB method which can reduce processor workload by avoiding unnecessary updates. The deadlines and periods of update transactions are assumed to have been assigned in the above works.

Various deadline and period deriving methods for maintaining the temporal consistency are proposed in recent two decades. Xiong and Ramamritham [11] proposed the More-Less scheme in which all transactions are released periodically. Xiong *et al.* [20] proposed the DS-FP method which reduces processor workload by judiciously deferring the sampling times of update transaction jobs. Han *et al.* [21] proposed the DS-EDF method which extends DS-FP to EDF scheduling environments. Jha *et al.* [22] proposed the first deadline and period deriving method which can guarantee the mutual temporal consistency of real-time data objects. Han *et al.* [23] proposed two online scheduling switch schemes, SBS and ABS, to search for the proper switch point. All above methods can be used in the environments with only update transactions or control transactions.

Recently, the co-scheduling of hybrid transactions becomes a hot research topic in RTDBS. Han *et al.* [24] proposed the AEDF-Co method which can be used in the systems with EDF scheduling. Wang *et al.* [25] proposed the PCS method that adopts a Fix Priority (FP) assignment scheme. Han *et al.* [8] proposed the Co-LALF method in which the release times of update jobs are deferred for reducing the process workload. All the above three methods aim to maximize the quality of data under the premise of the schedulability of control transactions, thus update transactions may miss their deadlines. Moreover, the above three works focus on the uniprocessor environments and their proposed methods cannot be used in multiprocessor systems.

Lundberg [26] proposed the first update transaction scheduling method for multiprocessor environments. Li *et al.* [27] considered the processor workload and proposed an EDF-based update transaction scheduling method which can minimize the processor workload under the premise of the temporal consistency of real-time data objects. Li *et al.* [28] extended their previous works and proposed a DM-based update transaction scheduling method. Wang *et al.* [29] considered jitter-based transactions and proposed the JB-ML and SJB-ML schemes by extending the ML method. Although the above three methods are effective in maintaining the validity of real-time data, they ignore the impact on the performance of the control transactions.

As described above, all existing methods focus on the uniprocessor co-scheduling or the update transaction scheduling, while this paper is the first work on the multiprocessor co-scheduling of hybrid transactions.

III. BACKGROUND, NOTATIONS AND PROBLEM DEFINITION

In this section, we review the definition of the temporal validity of real-time data objects. Then, some useful notations are given, followed by the definition of the problem to be addressed in this work.

A. TEMPORAL VALIDITY FOR DATA FRESHNESS

Data objects in RTDBSs are defined for representing the current status of real-world entities. Since the states of real-world entities change continuously, the sampled values of data objects are valid only in their respective temporal intervals. Ramamritham *et al.* [9] defined the temporal validity of real-time data objects by validity intervals, as follows.

Definition 1 [9]: At time t, a real-time data object (x_i) is temporally valid if the sum of its latest sampling time ($r_{i,j}$) and its validity interval length (V_i) is no smaller than t, i.e., $r_{i,j} + V_i \geq t$.

B. NOTATIONS AND ASSUMPTIONS

A RTDBS with $n = n^c + n^u$ transactions and m processors is considered in this work, where n^c and n^u are the numbers of control transactions and update transactions, respectively. Using $\mathcal{T}^c = \{\tau_1^c, \tau_2^c, \cdots, \tau_{n^c}^c\}$ to denote the control transaction set, τ_i^c is the i-th control transaction in $\mathcal{T}^c$. Similarly, using $\mathcal{T}^u = \{\tau_1^u, \tau_2^u, \cdots, \tau_{n^u}^u\}$ to denote the update transaction set, τ_i^u is the i-th update transaction in $\mathcal{T}^u$. Moreover, $\mathcal{T}$ is used to represent the hybrid transaction set in RTDBSs, i.e. $\mathcal{T} = \mathcal{T}^u \bigcup \mathcal{T}^c$.

Each control transaction $\tau_i^c \in \mathcal{T}^c$ can be characterized by a 3-tuple: $< C_i^c, D_i^c, T_i^c >$, where C_i^c is the Worst-Case Execution Time (WCET), D_i^c is the relative deadline and T_i^c is the period. Since the deadline-constrained transaction is considered in this work, we have $D_i^c \leq T_i^c$ for each τ_i^c. Moreover, using $U_i^c = C_i^c/T_i^c$ and $\delta_i^c = C_i^c/D_i^c$ to represent the utilization and the density of τ_i^c, respectively, the total utilization of $\mathcal{T}^c$, $U(\mathcal{T}^c)$, can be obtained by $U(\mathcal{T}^c) = \sum_{\tau_i^c \in \mathcal{T}^c} U_i^c$, and the maximum density of the control transactions in $\mathcal{T}^c$, $\delta_{max}(\mathcal{T}^c)$, can be obtained by $\delta_{max}(\mathcal{T}^c) = \max_{\tau_i^c \in \mathcal{T}^c} \{\delta_i^c\}$.

Similarly, each update transaction τ_i^u can also be characterized by a 3-tuple: $< C_i^u, D_i^u, T_i^u >$, where C_i^u is the WCET, D_i^u is the relative deadline and T_i^u is the period. Moreover, $D_i^u \leq T_i^u$ for each update transaction τ_i^u. Each update transaction τ_i^u can generate an infinite number of jobs. Using x_i to denote the data object maintaining by τ_i^u and using $\tau_{i,j}^u$ to denote the j-th job generated by τ_i^u, the release time of $\tau_{i,j}^u$ is the j-th sampled time of x_i and the finish time of $\tau_{i,j}^u$ is the j-th update time of x_i. Since the value of each data object x_i sampled at time t is valid only in $[t, t + V_i)$ (based on Definition 1), it is required that $D_i^u + T_i^u \leq V_i$ for each update transaction τ_i^u. To minimize the workload of processors, we require that $D_i^u + T_i^u = V_i$. Using $\lambda_i^u = C_i^u/V_i$ to represent the density factor of each update transaction τ_i^u, the maximum density factor of all update transactions in $\mathcal{T}^u$, denoted by $\lambda_{max}(\mathcal{T}^u)$, can be obtained by $\lambda_{max}(\mathcal{T}^u) = \max_{\tau_i^u \in \mathcal{T}^u} \{\lambda_i^u\}$, and the total density factor of $\mathcal{T}^u$, denoted by $\lambda(\mathcal{T}^u)$, can be calculated by $\lambda(\mathcal{T}^u) = \sum_{\tau_i^u \in \mathcal{T}^u} \lambda_i^u$. Notations U_i^u is used to denote the utilization of τ_i^u, if the deadlines and periods of τ_i^u have been assigned. The total utilization of $\mathcal{T}^u$, denoted by $U(\mathcal{T}^u)$, can be obtained by $U(\mathcal{T}^u) = \sum_{\tau_i^u \in \mathcal{T}^u} U_i^u$.

Finally, we assume that control transactions are sorted by the increasing order of their relative deadlines and update transactions are sorted by Shortest Validity First (SVF) [12] order, i.e., $D_i^c \leq D_j^c$ and $V_i \leq V_j$ if $i < j$. Moreover, using M_i ($1 \leq i \leq m$) to denoted the i-th processor, $\mathcal{T}^c(M_i)$ and $\mathcal{T}^u(M_i)$ are the control transaction set and update transaction set executed on M_i, respectively, and $\mathcal{T}(M_i) = \mathcal{T}^c(M_i) \bigcup \mathcal{T}^u(M_i)$ is the transaction set executed on M_i.

C. PROBLEM DEFINITION

To distinguish from the *traditional* real-time scheduling problem and the co-scheduling problem of hybrid transactions on uniprocessor platforms, we make the following definition.

Multiprocessor Scheduling of Hybrid Transactions (MSHT): Given a multiprocessor system $\mathcal{MP}$ and a hybrid transaction set $\mathcal{T} = \mathcal{T}^c \bigcup \mathcal{T}^u$, how to assign the deadline and the period for each update transaction (in $\mathcal{T}^u$) and schedule the transactions in $\mathcal{T}$, such that:

1) $D_i^u + T_i^u = V_i$ for each update transaction τ_i^u in $\mathcal{T}^u$.
2) $\mathcal{T}$ is schedulable under the used scheduling method.
3) The workload of processors is minimized.

where D_i^u and T_i^u are the deadline and the period of τ_i^u, respectively, and V_i is the valid interval length of the data object maintained by τ_i^u.

IV. TWO INTUITIVE SCHEDULING METHODS

In this section, we give two intuitive scheduling methods, named HH-based Partitioned scheduling (HH-P) and Partitioned scheduling for Hybrid Transactions (P-HT) respectively, for addressing the MSHT problem.

A. THE HH-P METHOD

HH [10] is a deadline and period deriving method under which the deadline and the period of each update transaction τ_i^u are always set to $V_i/2$. By combining HH and EDF, we can get an intuitive partitioned co-scheduling method, HH-P, for hybrid transaction sets in multiprocessor systems. Note that, the schedulability analysis is necessary in partitioned scheduling methods, which is used to control the assignment of transactions on processors. The following theorem gives a sufficient and necessary condition of EDF-schedulable transaction sets.

Theorem 1 [30], [31]: A real-time task set $\mathcal{T} = \{\tau_i, \ldots, \tau_n\}$ is EDF-schedulable in uniprocessor systems if and only if $U(\mathcal{T}) \leq 1$ and

$$\forall t \in S, h(t) = \sum_{i=1}^{n} DBF(\tau_i, t) \leq t \tag{1}$$

where

$$DBF(\tau_i, t) = (\lfloor (t - D_i)/T_i \rfloor + 1)C_i \tag{2}$$

$$S = \{d_k | d_k = kT_i + D_i \wedge d_k \le \min\{L_a, L_b\}, k \in \mathbb{N}\} \tag{3}$$

$$L_a = \max\left\{D_1, \ldots, D_n, \frac{\sum_{i=1}^{n}(T_i - D_i)U_i}{1 - U}\right\} \tag{4}$$

and L_b *is the final convergent value of* $w^{k+1} = w^k$ *satisfying*

$$w^0 = \sum_{i=1}^{n} C_i \tag{5}$$

$$w^{k+1} = \sum_{i=1}^{n} \left\lceil \frac{w^k}{T_i} \right\rceil \cdot C_i \tag{6}$$

Note that, Theorem 1 is inefficient because $h(t)$ (in Eq. (1)) needs to be calculated at each d_k in S satisfying Eq. (3). Zhang and Burns [31] proposed the QPA method that improves the efficiency by reducing the number of checking points. Clearly, QPA can be used in HH-P to control the assignment of transactions on processors.

Note that, although HH-P is an effective partitioned co-scheduling method for hybrid transactions in multiprocessor environments, the workload of processors is always heavy under HH-P. The following example shows this problem.

Example 1: Consider a hybrid transaction set $\mathcal{T} = \{\tau_1^c, \tau_2^c, \tau_1^u\}$ *executed in a 2-processor system and let* $\tau_1^c =< 1, 5, 6 >$, $\tau_2^c =< 3, 5, 6 >$, $C_1^u = 2$ *and* $V_1 = 16$. *Based on the above descriptions, we have that* $D_1^u = T_1^u = V_1/2 = 8$ *under* **HH-P**, *and all transactions will be assigned to the first processor. Moreover, the total workload of processors is equal to* $\frac{C_1^c}{T_1^c} + \frac{C_2^c}{T_2^c} + \frac{C_1^u}{T_1^u} = \frac{1}{6} + \frac{3}{6} + \frac{2}{8} = \frac{11}{12}$. *However, we find that the period of* τ_1^u *can be assigned to 14 if we assign* τ_1^u *to the second processor, and thus the total workload of processors can be reduced to* $\frac{1}{6} + \frac{3}{6} + \frac{2}{14} = \frac{17}{21}$.

The above example shows that there are serious drawbacks in terms of processor workload if we simply extend traditional uniprocessor co-scheduling methods to multiprocessor environments. The reason is that the goal of traditional uniprocessor co-scheduling methods is to derive deadlines and periods of update transactions and maximize the number of schedulable transactions, while multiprocessor co-scheduling needs to consider the balance of the workloads of processors to minimize the total workload of processors. Moreover, we also find that the total workload of processors can be reduced if we reassign the deadlines and periods of update transactions after the assignment of transactions on processors.

B. THE P-HT METHOD

In this section, we give another partitioned scheduling method, P-HT, which can reduce the total workload of processors of HH-P by relaxing the schedulability condition and reassigning the deadlines and periods of update transactions after the assignment of transactions on processors. Before showing it, we give a useful definition as follows.

Definition 2: Given an update transaction τ_i^u *and a time interval with length* t, **UDBF**(τ_i^u, t) *is the upper demand bound function of* τ_i^u, *which satisfies*

$$UDBF(\tau_i^u, t) = \begin{cases} 0, & 0 \le t < C_i^u \\ C_i^u, & C_i^u \le t < \frac{V_i}{2} \\ \left\lfloor \frac{t}{V_i/2} \right\rfloor C_i^u, & \frac{V_i}{2} \le t \end{cases} \tag{7}$$

Clearly, since $D_i^u \le T_i^u$ for each τ_i^u, $UDBF(\tau_i^u, t)$ is an upper bound of the workload of τ_i^u in any time interval with length t and beginning at a release time of τ_i^u. Based on Definition 2, we have the following lemma.

Lemma 1: Given an integer t *and an update transaction* τ_i^u, *it always holds that* $DBF(\tau_i^u, t) \le UDBF(\tau_i^u, t)$, *where* $DBF(\tau_i^u, t)$ *and* $UDBF(\tau_i^u, t)$ *satisfy Eqs. (2) and (7), respectively.*

Proof: Since $D_i^u + T_i^u = V_i$, we have $DBF(\tau_i^u, t) = (\left\lfloor \frac{t - D_i^u}{V_i - D_i^u} \right\rfloor + 1)C_i^u$ based on Eq. (2). Next, we discuss this lemma in the following three cases.

- $0 \le t < C_i^u$. Since $0 < t < C_i^u$ and $C_i^u \le D_i^u \le T_i^u$, we have $-T_i \le -D_i^u < t - D_i^u < C_i^u - D_i^u < 0$. Based on Eq. (2), we can get that $DBF(\tau_i^u, t) = (\left\lfloor \frac{t - D_i^u}{T_i^u} \right\rfloor + 1)C_i^u = (-1 + 1)C_i^u = 0$. Since $\text{UDBF}(\tau_i^u, t) = 0$ (based on Eq. (7)), we can derive that $\text{DBF}(\tau_i^u, t) = UDBF(\tau_i^u, t)$ in this case.
- $C_i^u \le t < \frac{V_i}{2}$. Since $t < \frac{V_i}{2}$ and $V_i > 0$, we have $t - D_i^u < \frac{V_i}{2} - D_i^u < V_i - D_i^u$. By $V_i - D_i^u = T_i^u > 0$, we have $\frac{t - D_i^u}{V_i - D_i^u} < 1$, which means $DBF(\tau_i^u, t) = (\left\lfloor \frac{t - D_i^u}{V_i - D_i^u} \right\rfloor + 1)C_i^u \le (0 + 1)C_i^u \le C_i^u$. Since $\text{UDBF}(\tau_i^u, t) = C_i^u$ (based on Eq. (7)), we can obtain that $\text{DBF}(\tau_i^u, t) \le UDBF(\tau_i^u, t)$ in this case.
- $\frac{V_i}{2} \le t$. First, we consider the case in which $\frac{V_i}{2} \le t < V_i$. Since $D_i \le \frac{V_i}{2} \le t < V_i$, we have $0 < \frac{t - D_i^u}{V_i - D_i^u} < 1$, which means $DBF(\tau_i^u, t) = (\left\lfloor \frac{t - D_i^u}{V_i - D_i^u} \right\rfloor + 1)C_i^u = C_i^u$. Moreover, by $\text{UDBF}(\tau_i^u, t) = \left\lfloor \frac{t}{V_i/2} \right\rfloor C_i^u = C_i$ (based on Eq. (7)), we have $DBF(\tau_i^u, t) = UDBF(\tau_i^u, t)$. Next, we consider the case in which $V_i \le t$. Since $V_i \le t$ and $V_i - D_i^u = T_i^u > 0$, $\frac{t - D_i^u}{V_i - D_i^u}$ increases with the growth of D_i^u. Since $D_i^u \le \frac{V_i}{2}$, we have $\frac{t - D_i^u}{V_i - D_i^u} \le \frac{t - V_i/2}{V_i - V_i/2} \le \frac{t}{V_i/2} - 1$. Based on Eqs. (2) and (7), we can get $DBF(\tau_i^u, t) = (\left\lfloor \frac{t - D_i^u}{V_i - D_i^u} \right\rfloor + 1)C_i^u \le \left\lfloor \frac{t}{V_i/2} \right\rfloor C_i^u \le UDBF(\tau_i^u, t)$. Since $DBF(\tau_i^u, t) \le UDBF(\tau_i^u, t)$ in both cases $\frac{V_i}{2} \le t < V_i$ and $V_i \le t$, we can get that $DBF(\tau_i^u, t) \le UDBF(\tau_i^u, t)$ if $\frac{V_i}{2} \le t$.

Based on the above discussions, we have $DBF(\tau_i^u, t) \le UDBF(\tau_i^u, t)$ and the proof is thus finished. □

Lemma 1 shows the relation of $UDBF(\tau_i^u, t)$ and $DBF(\tau_i^u, t)$. Based on Lemma 1 and Theorem 1, a sufficient condition of EDF-schedulable hybrid transaction sets can be derived, as follows.

Theorem 2: A hybrid transaction set $\mathcal{T} = \{\tau_1^c, \ldots, \tau_{n^c}^c, \tau_1^u, \ldots, \tau_{n^u}^u\}$ *is EDF-schedulable in uniprocessor systems after executing* **HH** *on it if*

$$U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) \leq 1 \tag{8}$$

and

$$\forall t \in Q, \sum_{i=1}^{n^c} DBF(\tau_i^c, t) + \sum_{i=1}^{n^u} UDBF(\tau_i^u, t) \leq t, \tag{9}$$

where $\mathcal{T}^c = \{\tau_1^c, \ldots, \tau_{n^c}^c\}$, $\mathcal{T}^u = \{\tau_1^u, \ldots, \tau_{n^u}^u\}$, $DBF(\tau_i^c, t)$ *and* $UDBF(\tau_i^u, t)$ *satisfy Eqs. (2) and (7), respectively,* $Q = Q_1 \bigcup Q_2$,

$$Q_1 = \{d_k \,|\, d_k = kT_i^c + D_i^c \wedge d_k < \min\{\overline{L_a}, \overline{L_b}\}, k \in N\} \tag{10}$$

$$Q_2 = \{e_k \,|\, e_k = k\frac{V_i}{2} \wedge e_k < \min\{\overline{L_a}, \overline{L_b}\}, k \in N\}\mathcal{P}\} \tag{11}$$

$$\overline{L_a} = \max\{D_1^c, \cdots, D_{n^c}^c, V_1/2, \cdots, V_{n^u}/2, \mathcal{P}\} \tag{12}$$

$$\mathcal{P} = \frac{\sum_{i=1}^{n^c}(T_i^c - D_i^c)U_i^c + \sum_{j=1}^{n^u} C_j^u}{1 - U(\mathcal{T}^c) - 2\lambda(\mathcal{T}^u)} \tag{13}$$

and $\overline{L_b}$ *is the final convergent value of* $\overline{w}^{k+1} = \overline{w}^k$ *satisfying*

$$\overline{w}^0 = \sum_{i=1}^{n^c} C_i^c + \sum_{j=1}^{n^u} C_j^u \tag{14}$$

$$\overline{w}^{k+1} = \sum_{i=1}^{n^c} \left\lceil \frac{\overline{w}^k}{T_i^c} \right\rceil C_i^c + \sum_{j=1}^{n^u} \left\lceil \frac{\overline{w}^k}{V_j/2} \right\rceil C_j^u \tag{15}$$

Proof: Use $\overline{\mathcal{T}}^u$ and $\overline{\mathcal{T}}$ to represent the update transaction set and the transaction set after executing **HH** on $\mathcal{T}$, respectively. It can be obtained that $D_i^u = T_i^u = V_i/2$ for each update transaction τ_i^u in $\overline{\mathcal{T}}^u$. Since the deadlines and periods of update transactions have be assigned after the execution of **HH**, update transactions in $\overline{\mathcal{T}}^u$ can be treated as control transactions. So, we only need to prove that $\overline{\mathcal{T}}$ must satisfy the conditions in Theorem 1 if $\mathcal{T}$ satisfies the conditions in this Theorem.

First, since $\lambda_i^u = C_i^u/V_i$ and $U_i^u = C_i^u/T_i^u$, as well as $T_i^u = V_i/2$, we have $2\lambda_i^u = U_i^u$. By Eq.(8), we have $U(\overline{\mathcal{T}}) = U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) \leq 1$.

Next, we consider the relation between Eqs. (1) and (9). Note that, $D_i^u = T_i^u = V_i/2$ for each update transaction τ_i^u in $\overline{\mathcal{T}}^u$. So, based on Eqs. (5), (6), (14) and (15), we have $\overline{L_b} = L_b$ for $\overline{\mathcal{T}}$, where L_b satisfies the conditions in Theorem 1. Moreover, by $U(\overline{\mathcal{T}}^u) = U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u)$ and $D_i^u > 0$, we have $\mathcal{P} > \frac{\sum_{i=1}^{n^c}(T_i^c - D_i^c)U_i^c + \sum_{j=1}^{n^u}(T_j^u - D_j^u)U_j^u}{1 - U(\overline{\mathcal{T}})}$, which means $\overline{L_a} > L_a$ for $\overline{\mathcal{T}}$, where L_a satisfies Eq. (4). Since $\overline{L_b} = L_b$, we have $\min\{\overline{L_a}, \overline{L_b}\} > \min\{L_a, L_b\}$, which means $S \subseteq Q$ for $\overline{\mathcal{T}}$, where S satisfies Eq. (3). Note that, $UDBF(\tau_i^u, t) \geq DBF(\tau_i^u, t)$ for each update transaction τ_i^u (based on Lemma 1). So, by Eq. (9), we have

Algorithm 1 P-HT

```
input : A hybrid transaction set
        $\mathcal{T} = \{\tau_1^u, \ldots, \tau_{n^u}^u, \tau_1^c, \ldots, \tau_{n^c}^c\}$; a processor set
        $M = \{M_1, \ldots, M_m\}$.
output: A schedule $\mathcal{S}$.
1  begin
2      for $i = 1; i \leq n^u; i++$ do
3          Execute HH to set both $D_i^u$ and $T_i^u$ to $\frac{V_i}{2}$;
4      Sort all transactions by their increasing deadlines to
       transform $\mathcal{T}$ to $\mathcal{T}' = \{\tau_1, \ldots, \tau_{n^u+n^c}\}$;
5      for $i = 1; i \leq n^u + n^c; i++$ do
6          for $j = 1; j \leq m; j++$ do
7              Assign $\tau_i$ to $M_j$;
8              if The transactions on $M_j$ satisfy the
               conditions in Theorem 2 then
9                  Break;
10             Remove $\tau_i$ from $M_j$;
11         if $j == m + 1$ then
12             Return Fail;
13     Execute MDC to derive the deadlines and periods of
       the update transactions allocated on each processor;
14     Schedule the transactions on each processor by EDF
       to get the schedule $\mathcal{S}$;
15     Return $\mathcal{S}$;
```

$$\sum_{\tau_i \in \overline{\mathcal{T}}} DBF(\tau_i, t) \leq \sum_{i=1}^{n^c} DBF(\tau_i^c, t) + \sum_{i=1}^{n^u} UDBF(\tau_i^u, t) \leq 1.$$

Since $U(\overline{\mathcal{T}}) = U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) \leq 1$, we can obtain that $\overline{\mathcal{T}}$ is schedulable under EDF based on Theorem 1. The proof is thus finished. □

Theorem 2 gives a relax schedulability condition for hybrid transaction sets. MDC [32] is a deadline and period deriving method under which the workload is always minimized. Therefore, by combing the theories of Theorem 2 and MDC, we can derive an effective partitioned scheduling method, named by Partitioned scheduling for Hybrid Transactions (**P-HT**), for hybrid transactions in multiprocessor systems.

Algorithm 1 shows the pseudo-code of **P-HT**. First, HH is used to initialize the deadlines of update transactions (line 3) and all transactions are sorted by increasing order of their deadlines (line 4). Then, transactions are assigned to processors based on Theorem 2 (lines 5-10). If there is a transaction that cannot be dispatched, we return "Fail" (line 12). Otherwise, the MDC method is executed to reassign the deadlines (and periods) for update transactions (line 13). Finally, EDF is invoked to schedule the transactions on each processor (line 14) and the schedule is returned (line 15).

Obviously, **P-HT** can reduce the total workload of processors under **HH-P**, since the periods of update transactions may be extended after the reassignment (line 13). Actually, the strategy of setting the period of τ_1^u to 14 and assigning τ_1^u to the second processor in Example 1 is the result of executing **P-HT**.

V. IMPROVED P-HT METHODS

In this section, we propose two scheduling methods, named Efficient Partitioned scheduling for Hybrid Transactions (EP-HT) and Improved EP-HT scheduling (IEP-HT), respectively. EP-HT improves the efficiency of P-HT and IEP-HT improves the acceptance ratio of EP-HT.

A. THE EP-HT METHOD

Before showing the EP-HT method, we give a useful definition as follows.

Definition 3: Given an update transaction τ_i^u and an integer t, UDBF$^*(\tau_i^u, t)$ is the approximate upper demand bound function of τ_i^u, which satisfies

$$UDBF^*(\tau_i^u, t) = \begin{cases} 0, & 0 \le t < C_i^u \\ C_i^u, & C_i^u \le t < \dfrac{V_i}{2} \\ \dfrac{t}{V_i/2} C_i^u, & \dfrac{V_i}{2} \le t \end{cases} \tag{16}$$

Based on Definition 3, we have the following lemma.

Lemma 2: For any update transaction τ_i^u and integer t, $t \ge 0$, it always holds that

$$UDBF(\tau_i^u, t) \le UDBF^*(\tau_i^u, t) < 2UDBF(\tau_i^u, t) \tag{17}$$

where $UDBF(\tau_i^u, t)$ and $UDBF^*(\tau_i^u, t)$ satisfy Eqs. (7) and (16), respectively.

Proof: By Eqs. (7) and (16), we can get that $UDBF(\tau_i^u, t) = UDBF^*(\tau_i^u, t)$ if $0 < t < V_i/2$. Since $UDBF(\tau_i^u, t) \ge 0$, we have $UDBF(\tau_i^u, t) = UDBF^*(\tau_i^u, t) < 2\ UDBF(\tau_i^u, t)$ if $0 < t < V_i/2$. Next, we consider the case in which $V_i/2 \le t$. Since $V_i/2 \le t$, we have $\frac{t}{V_i/2} \ge 1$, which means $\left\lfloor \frac{t}{V_i/2} \right\rfloor \le \frac{t}{V_i/2} < 2\left\lfloor \frac{t}{V_i/2} \right\rfloor$. By Eqs. (7) and (16), we have $UDBF(\tau_i^u, t) \le UDBF^*(\tau_i^u, t) < 2UDBF(\tau_i^u, t)$ if $V_i/2 \le t$.

Since $UDBF(\tau_i^u, t) \le UDBF^*(\tau_i^u, t) < 2UDBF(\tau_i^u, t)$ in both cases $0 < t < \frac{V_i}{2}$ and $\frac{V_i}{2} \le t$, we can get Eq. (17). The proof is thus finished. □

Based on Lemma 2 and Theorem 2, two important theorems can be obtained, as follows.

Theorem 3: Given a control transaction $\tau_{n^c}^c$ and an EDF-schedulable hybrid transaction set $\mathcal{T}' = \{\tau_1^c, \ldots, \tau_{n^c-1}^c, \tau_1^u, \ldots, \tau_{n^u}^u\}$, let $\mathcal{T} = \mathcal{T}' \bigcup \{\tau_{n^c}^c\}$, there must be some deadline and period deriving methods under which $\mathcal{T}$ is EDF-schedulable if

$$D_{n^c}^c \ge \max\{D_1^c, \ldots, D_{n^c-1}^c, V_1/2, \ldots, V_{n^u}/2\} \tag{18}$$

and

$$D_{n^c}^c - \Delta^*(D_{n^c}^c, \mathcal{T}') \ge C_{n^c}^c \tag{19}$$

where

$$\Delta^*(t, \mathcal{T}') = \sum_{\tau_i^c \in \mathcal{T}'^c} DBF^*(\tau_i^c, t) + \sum_{\tau_i^u \in \mathcal{T}'^u} UDBF^*(\tau_i^u, t) \tag{20}$$

$$DBF^*(\tau_i, t) = \begin{cases} 0, & t < D_i \\ C_i + (t - D_i)U_i, & \text{otherwise} \end{cases} \tag{21}$$

$\mathcal{T}'^c = \{\tau_1^c, \ldots, \tau_{n^c-1}^c\}$, $\mathcal{T}'^u = \{\tau_1^u, \ldots, \tau_{n^u}^u\}$ and $UDBF^*(\tau_i^u, t)$ satisfies Eq. (16).

Proof: First, let $\mathcal{T}^c = \mathcal{T}'^c \bigcup \{\tau_{n^c}^c\}$ and $\mathcal{T}^u = \mathcal{T} - \mathcal{T}^c = \mathcal{T}'^u$, we prove that $U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) \le 1$ if $\mathcal{T}$ satisfies both Eqs. (18) and (19). Based on Eqs. (16) and (18), we have $UDBF^*(\tau_i^u, D_{n^c}^c) = \frac{D_{n^c}^c}{V_i/2} C_i^u = 2\lambda_i^u D_{n^c}^c$ for each τ_i^u (in $\mathcal{T}'^u$). Based on Eqs. (18) and (21), we have $DBF^*(\tau_i^c, D_{n^c}^c) = C_i^c + (D_{n^c}^c - D_i^c)U_i^c \ge D_{n^c}^c U_i^c$ for each $\tau_i^c \in \mathcal{T}'^c$. Therefore, we can derive that $\Delta^*(D_{n^c}^c, \mathcal{T}') \ge \sum_{\tau_i^c \in \mathcal{T}'^c} D_{n^c}^c U_i^c + \sum_{\tau_i^u \in \mathcal{T}'^u} 2\lambda_i^u D_{n^c}^c \ge D_{n^c}^c(U(\mathcal{T}'^c) + 2\lambda(\mathcal{T}'^u))$ based on Eq. (20), which means $D_{n^c}^c - \Delta^*(D_{n^c}^c, \mathcal{T}') \le D_{n^c}^c(1 - U(\mathcal{T}'^c) - 2\lambda(\mathcal{T}'^u))$. By Equation (20), we have $C_{n^c}^c \le D_{n^c}^c(1 - U(\mathcal{T}'^c) - 2\lambda(\mathcal{T}'^u))$. Note that, $U_{n^c}^c = C_{n^c}^c/T_{n^c}^c \le C_{n^c}^c/D_{n^c}^c$. So, we have $U_{n^c}^c \le 1 - U(\mathcal{T}'^c) - 2\lambda(\mathcal{T}'^u)$, which means

$$U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) \le 1 \tag{22}$$

Next, assuming that $\mathcal{T}$ is unschedulable under EDF and using t^* to denote the first time unit at which there appear some expired transactions, we prove this theorem by contradiction. Since some transactions miss their deadlines at t^*, we have $\sum_{\tau_i^c \in \mathcal{T}^c} DBF(\tau_i^c, t^*) + \sum_{\tau_i^u \in \mathcal{T}^u} DBF(\tau_i^u, t^*) > t^*$, where $DBF(\cdot)$ satisfies Eq. (2). Based on Lemma 1, we have $\sum_{\tau_i^c \in \mathcal{T}^c} DBF(\tau_i^c, t^*) + \sum_{\tau_i^u \in \mathcal{T}^u} UDBF(\tau_i^u, t^*) \ge \sum_{\tau_i^c \in \mathcal{T}^c} DBF(\tau_i^c, t^*) + \sum_{\tau_i^u \in \mathcal{T}^u} DBF(\tau_i^u, t^*) > t^*$. Note that, $DBF^*(\tau_i^c, t^*) \ge DBF(\tau_i^c, t^*)$ for each control transaction τ_i^c. So, based on Lemma 2 and Eq. (20), we have

$$\begin{aligned} &\Delta^*(t^*, \mathcal{T}) \\ &\quad = \sum_{\tau_i^c \in \mathcal{T}^c} DBF^*(\tau_i^c, t) + \sum_{\tau_i^u \in \mathcal{T}^u} UDBF^*(\tau_i^u, t) \\ &\quad \ge \sum_{\tau_i^c \in \mathcal{T}^c} DBF(\tau_i^c, t^*) + \sum_{\tau_i^u \in \mathcal{T}^u} UDBF(\tau_i^u, t^*) \\ &\quad > t^* \end{aligned} \tag{23}$$

Note that, $\mathcal{T}'$ is EDF-schedulable, $\mathcal{T} = \mathcal{T}' \bigcup \{\tau_{n^c}^c\}$ and t^* is the first time unit at which there appear some expired transactions. So, we have $t^* \ge D_{n^c}^c$. Based on Eqs. (16), (18), (20) and (21), we have

$$\begin{aligned} &\Delta^*(t^*, \mathcal{T}) \\ &= \sum_{\tau_i^c \in \mathcal{T}^c} DBF^*(\tau_i^c, t^*) + \sum_{\tau_i^u \in \mathcal{T}^u} UDBF^*(\tau_i^u, t^*) \\ &= \sum_{\tau_i^c \in \mathcal{T}^c} (C_i^c + U_i^c(t^* - D_i^c)) + \sum_{\tau_i^u \in \mathcal{T}^u} (\frac{t^*}{V_i/2} C_i^u) \\ &= \sum_{\tau_i^c \in \mathcal{T}^c} (C_i^c + U_i^c(D_{n^c}^c - D_i^c + t^* - D_{n^c}^c)) + \sum_{\tau_i^u \in \mathcal{T}^u} (\frac{t^*}{V_i/2} C_i^u) \\ &= \Delta^*(D_{n^c}^c, \mathcal{T}) + \sum_{\tau_i^c \in \mathcal{T}^c} (U_i^c(t^* - D_{n^c}^c)) + \sum_{\tau_i^u \in \mathcal{T}^u} (\frac{t^* - D_{n^c}^c}{V_i/2} C_i^u) \\ &= \Delta^*(D_{n^c}^c, \mathcal{T}) + (t^* - D_{n^c}^c)(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u)) \end{aligned} \tag{24}$$

By Eqs. (23) and (24), we have

$$\Delta^*(D^c_{n^c}, \mathcal{T}) + (t^* - D^c_{n^c})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u)) - t^* > 0 \quad (25)$$

Note that, $\Delta^*(D^c_{n^c}, \mathcal{T}) = \Delta^*(D^c_{n^c}, \mathcal{T}') + DBF^*(\tau^c_{n^c}, D^c_{n^c}) = \Delta^*(D^c_{n^c}, \mathcal{T}')+C^c_{n^c}$ based on Eqs. (20) and (21). By Eq. (19), we have

$$\begin{aligned}&\Delta^*(D^c_{n^c}, \mathcal{T}) + (t^* - D^c_{n^c})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u)) - t^*\\ &= \Delta^*(D^c_{n^c}, \mathcal{T}') + C^c_{n^c} - t^* + (t^* - D^c_{n^c})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u))\\ &\le D^c_{n^c} - t^* + (t^* - D^c_{n^c})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u))\\ &\le (t^* - D^c_{n^c})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) - 1) \qquad (26)\end{aligned}$$

Based on Eqs. (25) and (26), we have $(t^* - D^c_{n^c})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) - 1) > 0$. Since $t^* \ge D^c_{n^c}$, we have $U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) > 1$. Clearly, it is contradict with Eq. (22), which means the assumption cannot hold and there must be some deadline and period deriving methods which can guarantee the schedulability of $\mathcal{T}$. The proof is thus finished. □

Theorem 3 gives a sufficient condition of EDF-schedulable hybrid transaction sets if the newly inserted transaction is a control transaction. Next, we consider the case in which the newly inserted transaction is an update transaction.

Theorem 4: Given an update transaction $\tau^u_{n^u}$ and an EDF-schedulable hybrid transaction set $\mathcal{T}' = \{\tau^c_1, \ldots, \tau^c_{n^c}, \tau^u_1, \ldots, \tau^u_{n^u-1}\}$, let $\mathcal{T} = \mathcal{T}' \bigcup \{\tau^u_{n^u}\}$, there must be some deadline and period deriving methods under which $\mathcal{T}$ is EDF-schedulable if

$$V^u_{n^u} \ge \max\{2D^c_1, \ldots, 2D^c_{n^c}, V_1, \ldots, V_{n^u-1}\} \qquad (27)$$

and

$$V_{n^u}/2 - \Delta^*(V_{n^u}/2, \mathcal{T}') \ge C^u_{n^u} \qquad (28)$$

where

$$\Delta^*(t, \mathcal{T}') = \sum_{\tau^c_i \in \mathcal{T}'^c} DBF^*(\tau^c_i, t) + \sum_{\tau^u_i \in \mathcal{T}'^u} UDBF^*(\tau^u_i, t) \qquad (29)$$

$\mathcal{T}'^c = \{\tau^c_1, \ldots, \tau^c_{n^c}\}$, $\mathcal{T}'^u = \{\tau^u_1, \ldots, \tau^u_{n^u-1}\}$, $DBF^*(\tau^c_i, t)$ and $UDBF^*(\tau^u_i, t)$ satisfy Eqs. (21) and (16), respectively.

Proof: Setting $D^u_{n^u} = T^u_{n^u} = V_{n^u}/2$ for each update transactions in $\mathcal{T}$, we prove that $\mathcal{T}$ is EDF-schedulable if it satisfies the conditions in this theorem.

First, let $\mathcal{T}^u = \mathcal{T}'^u \bigcup \{\tau^u_{n^u}\}$ and $\mathcal{T}^c = \mathcal{T} - \mathcal{T}^u = \mathcal{T}'^c$, we prove that $U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) \le 1$. Since $D^u_{n^u} = V_{n^u}/2$, we have $UDBF^*(\tau^u_i, D^u_{n^u}) = 2\lambda^u_i D^u_{n^u}$ for each $\tau^u_i \in \mathcal{T}'^u$, based on Eqs. (16) and (27). By Eqs. (21) and (27), we have $DBF^*(\tau^c_i, D^u_{n^u}) = C^c_i + (D^u_{n^u} - D^c_i)U^c_i \ge D^u_{n^u} U^c_i$ for each $\tau^c_i \in \mathcal{T}'^c$. So, by Eq. (29), we can get that

$$\begin{aligned}&\Delta^*(D^u_{n^u}, \mathcal{T}')\\ &= \sum_{\tau^c_i \in \mathcal{T}'^c} DBF^*(\tau^c_i, D^u_{n^u}) + \sum_{\tau^u_i \in \mathcal{T}'^u} UDBF^*(\tau^u_i, D^u_{n^u})\\ &\ge \sum_{\tau^c_i \in \mathcal{T}'^c} D^u_{n^u} U^c_i + \sum_{\tau^u_i \in \mathcal{T}'^u} 2\lambda^u_i D^u_{n^u}\\ &\ge D^u_{n^u}(U(\mathcal{T}'^c) + 2\lambda(\mathcal{T}'^u)) \qquad (30)\end{aligned}$$

Algorithm 2 EP-HT

```
input : A hybrid transaction set
        $\mathcal{T} = \{\tau^u_1, \ldots, \tau^u_{n^u}, \tau^c_1, \ldots, \tau^c_{n^c}\}$; a processor set
        $M = \{M_1, \ldots, M_m\}$.
output: A schedule $\mathcal{S}$.
1  begin
2      for $i = 1; i \le n^u; i++$ do
3          Execute HH to set both $D^u_i$ and $T^u_i$ to $\frac{V_i}{2}$;
4      Sort all transactions by their increasing deadlines to
       transform $\mathcal{T}$ to $\mathcal{T}' = \{\tau_1, \ldots, \tau_{n^u+n^c}\}$;
5      for $i = 1; i \le n^u + n^c; i++$ do
6          for $j = 1; j \le m; j++$ do
7              if $\tau_i$ is a control transaction in $\mathcal{T}'$ then
8                  Assign $\tau_i$ to $M_j$;
9                  if The transactions on $M_j$ satisfy the
                   conditions in Theorem 3 then
10                     Break;
11                 Remove $\tau_i$ from $M_j$;
12             else
13                 Assign $\tau_i$ to $M_j$;
14                 if The transactions on $M_j$ satisfy the
                   conditions in Theorem 4 then
15                     Break;
16                 Remove $\tau_i$ from $M_j$;
17         if $j == m + 1$ then
18             Return Fail;
19     Execute MDC to derive the deadlines and periods of
       the update transactions allocated on each processor;
20     Schedule the transactions on each processor by EDF
       to get the schedule $\mathcal{S}$;
21     Return $\mathcal{S}$;
```

Note that, $D^u_{n^u} = V_{n^u}/2$. Therefore, based on Eq. (28) and (30), we have

$$\begin{aligned}C^u_{n^c} &\le D^u_{n^u} - \Delta^*(D^u_{n^u}, \mathcal{T}')\\ &\le D^u_{n^u} - D^c_{n^c}(U(\mathcal{T}'^c) + 2\lambda(\mathcal{T}'^u))\\ &\le D^u_{n^u}(1 - U(\mathcal{T}'^c) - 2\lambda(\mathcal{T}'^u)), \qquad (31)\end{aligned}$$

which means

$$\begin{aligned}2\lambda^u_{n^u} &= 2C^u_{n^u}/V_{n^u}\\ &= C^u_{n^u}/D^u_{n^u}\\ &\le \frac{D^u_{n^u} - D^c_{n^c}(U(\mathcal{T}'^c) + 2\lambda(\mathcal{T}'^u))}{D^u_{n^u}}\\ &\le 1 - (U(\mathcal{T}'^c) + 2\lambda(\mathcal{T}'^u)) \qquad (32)\end{aligned}$$

Since $\mathcal{T}^u = \mathcal{T}'^u \bigcup \{\tau^u_{n^u}\}$ and $\mathcal{T}^c = \mathcal{T}'^c$, we have

$$U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) = U(\mathcal{T}'^c) + 2\lambda(\mathcal{T}'^u) + 2\lambda^u_{n^u} \le 1 \quad (33)$$

Next, we prove that $\mathcal{T}$ is EDF-schedulable, by contradiction. Assuming that $\mathcal{T}$ is unschedulable under EDF and t^* is the first time unit at which there appear some expired transactions. Clearly, it should be satisfied that $\sum_{\tau_i^c \in \mathcal{T}^c} DBF(\tau_i^c, t^*) + \sum_{\tau_i^u \in \mathcal{T}^u} DBF(\tau_i^u, t^*) > t^*$. Based on Lemma 1, we have $\sum_{\tau_i^c \in \mathcal{T}^c} DBF(\tau_i^c, t^*) + \sum_{\tau_i^u \in \mathcal{T}^u} UDBF(\tau_i^u, t^*) > t^*$. Note that, $DBF^*(\tau_i^c, t^*) \geq DBF(\tau_i^c, t^*)$ for each τ_i^c and $UDBF^*(\tau_i^u, t^*) \geq UDBF(\tau_i^u, t^*)$ for each τ_i^u (based on Lemma 2). So, by Eq. (29), we have

$$\begin{aligned}\Delta^*(t^*, \mathcal{T}) \\ &= \sum_{\tau_i^c \in \mathcal{T}'^c} DBF^*(\tau_i^c, t) + \sum_{\tau_i^u \in \mathcal{T}'^u} UDBF^*(\tau_i^u, t) \\ &\geq \sum_{\tau_i^c \in \mathcal{T}^c} DBF(\tau_i^c, t^*) + \sum_{\tau_i^u \in \mathcal{T}^u} UDBF(\tau_i^u, t^*) \\ &> t^* \end{aligned} \tag{34}$$

Note that, $\mathcal{T}'$ is schedulable under EDF and t^* is the first time unit at which there appear some expired transactions (in $\mathcal{T} = \mathcal{T}' \bigcup \{\tau_{n^u}^u\}$). Therefore, we have $t^* \geq D_{n^c}^u$. Since $D_{n^u}^u = V_{n^u}/2$, we have $t^* \geq V_{n^u}/2$. Based on Eqs. (16), (21), (27) and (29), we have

$$\begin{aligned}\Delta^*(t^*, \mathcal{T}) \\ &= \sum_{\tau_i^c \in \mathcal{T}^c} DBF^*(\tau_i^c, t^*) + \sum_{\tau_i^u \in \mathcal{T}^u} UDBF^*(\tau_i^u, t^*) \\ &= \sum_{\tau_i^c \in \mathcal{T}^c} (C_i^c + U_i^c(t^* - D_i^c)) + \sum_{\tau_i^u \in \mathcal{T}^u} (\frac{t^*}{V_i/2} C_i^u) \\ &= \Delta^*(\frac{V_{n^u}}{2}, \mathcal{T}) + \sum_{\tau_i^c \in \mathcal{T}^c} U_i^c(t^* - \frac{V_{n^u}}{2}) + \sum_{\tau_i^u \in \mathcal{T}^u} (\frac{t^* - V_{n^u}/2}{V_i/2} C_i^u) \\ &= \Delta^*(\frac{V_{n^u}}{2}, \mathcal{T}) + (t^* - \frac{V_{n^u}}{2})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u)) \end{aligned} \tag{35}$$

Based on Eqs. (34) and (35), we have

$$\Delta^*(\frac{V_{n^u}}{2}, \mathcal{T}) + (t^* - \frac{V_{n^u}}{2})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u)) - t^* > 0 \tag{36}$$

Note that, $\Delta^*(V_{n^u}/2, \mathcal{T}) = \Delta^*(V_{n^u}/2, \mathcal{T}') + UDBF^*(\tau_{n^u}^u, V_{n^u}/2) = \Delta^*(V_{n^u}/2, \mathcal{T}') + C_{n^u}^u \leq V_{n^u}/2$ based on Eqs. (16), (28) and (29). Therefore, we have

$$\begin{aligned}\Delta^*(\frac{V_{n^u}}{2}, \mathcal{T}) + (t^* - \frac{V_{n^u}}{2})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u)) - t^* \\ &\leq \frac{V_{n^u}}{2} - t^* + (t^* - \frac{V_{n^u}}{2})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u)) \\ &\leq (t^* - \frac{V_{n^u}}{2})(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) - 1) \end{aligned} \tag{37}$$

Based on Eqs. (36) and (37), we can get $(t^* - V_{n^u}/2)(U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) - 1) > 0$. Since $t^* \geq V_{n^c}/2$, we have $U(\mathcal{T}^c) + 2\lambda(\mathcal{T}^u) > 1$. Clearly, it is contradict with Eq. (33). Therefore, we have that the assumption cannot hold and there must be some deadline and period deriving methods under which $\mathcal{T}$ is EDF-schedulable. The proof is thus finished. □

Given an EDF-schedulable hybrid transaction set $\mathcal{T}$, Theorem 3 gives a sufficient condition to test the EDF-schedulability of the new transaction set generated by inserting a control transaction into $\mathcal{T}$, and Theorem 3 gives a sufficient condition to test the EDF-schedulability of the new transaction set generated by inserting an update transaction into $\mathcal{T}$. By using the theories of Theorems 3 and 4, we can get a novel scheduling method. Naming this method by **EP-HT**, the pseudo-code is shown in Algorithm 2. Clearly, different from **P-HT**, **EP-HT** uses Theorem3 (line 9) or Theorem 4 (line 14) to dispatch transactions based on the type of each transaction.

EP-HT has a higher efficiency than **P-HT** since **P-HT** needs to calculate Eq. (9) at each $t \in Q$. But, it should be pointed out that, the efficiency improvement is at the expense of acceptance ratio. So, it is necessary to improve the acceptance ratio of **EP-HT**.

B. THE IEP-HT METHOD

As shown in Algorithm 2, in **EP-HT**, the final deadlines and periods of update transactions are obtained after the assignment of transactions to processors. Since schedulability tests (Theroems 3 and 4) are executed under a pessimistic assumption that $D_i^u = T_i^u = V_i/2$ for each update transaction τ_i^u, **EP-HT** has a poor performance on acceptance ratio. Based on the above observation, we improve **EP-HT** and propose a new scheduling method, named **IEP-HT**, which calculates final deadlines and periods for update transactions once they are assigned to processors. Note that update transactions can be treated as control transactions (also real-time tasks) if their deadlines and periods have be assigned. Next, we review a useful theorem for testing the schedulability of a real-time task set.

Theorem 5 [33]: Given a real-time task τ_n and a real-time task set $\mathcal{T}' = \{\tau_1, \ldots, \tau_{n-1}\}$ which is schedulable under EDF on uniprocessor platforms, $\mathcal{T} = \mathcal{T}' \bigcup \{\tau_i\}$ is EDF-schedulable if $U(\mathcal{T}) \leq 1$ and

$$D_n - \sum_{i=1}^{n-1} DBF^*(\tau_i, D_n) \geq C_n \tag{38}$$

By using Theorem 5, we can get the **IEP-HT** method as shown in Algorithm 3. Whenever an update transaction τ_i^u is dispatched to a processor successfully, the MDC is used to calculate the deadline for τ_i^u (line 10). Compared with **EP-HT**, **IEP-HT** can dispatch more transactions to the processor.

C. THEORETICAL EVALUATIONS OF EP-HT AND IEP-HT

The partitioned multiprocessor real-time scheduling for sporadic real-time tasks has been recently studied by Baruah and Fisher in [34], Chen and Chakraborty in [35], and Fisher et al. in [36]. For update transaction, the partitioned multiprocessor real-time scheduling has been researched by Li et al. in [28]. Resource augmentation bounds [37] have been derived to quantify the worst-case performance of their partition schemes. In this work, similar to [28], [34]–[36], we also offer a quantitative evaluation of our algorithms in terms of

Algorithm 3 IEP-HT

```
input : A hybrid transaction set
        $\mathcal{T} = \{\tau_1^u, \ldots, \tau_{n^u}^u, \tau_1^c, \ldots, \tau_{n^c}^c\}$; a processor set
        $M = \{M_1, \ldots, M_m\}$.
output: A schedule $\mathcal{S}$.
1  begin
2      for $i = 1; i \leq n^u; i++$ do
3          Execute HH to set both $D_i^u$ and $T_i^u$ to $\frac{V_i}{2}$;
4      Sort all transactions by their increasing deadlines to
       transform $\mathcal{T}$ to $\mathcal{T}' = \{\tau_1, \ldots, \tau_{n^u+n^c}\}$;
5      for $i = 1; i \leq n^u + n^c; i++$ do
6          for $j = 1; j \leq m; j++$ do
7              Assign $\tau_i$ to $M_j$;
8              if The transactions on $M_j$ satisfy the
               conditions in Theorem 5 then
9                  if $\tau_i$ is a update transaction in $\mathcal{T}'$ then
10                     Execute MDC to assign deadline for
                       $\tau_i$;
11                 Break;
12             Remove $\tau_i$ from $M_j$;
13         if $j == m + 1$ then
14             Return Fail;
15     Schedule the transactions on each processor by EDF
       to get the schedule $\mathcal{S}$;
16     Return $\mathcal{S}$;
```

resource augmentation bound. But it should be noted that the problem we addressed is different from theirs in that hybrid transaction sets are considered in our research.

In the previous subsection, we can get that, Eq. (38) always holds if the newly inserted control transaction $\tau_{n^c}^c$ satisfies Eq. (19), and Eq. (38) always holds if the newly inserted update transaction $\tau_{n^u}^u$ satisfies Eq. (28), based on Theorems 3, 4 and 5. So, we have that the resource augmentation of IEP-HT must be smaller than or equal to that of EP-HT. Next, we give an improvement lemma which is useful for analyzing the resource augmentation of EP-HT, as follows.

Lemma 3: Give a hybrid transaction set $\mathcal{T} = \mathcal{T}^c \bigcup \mathcal{T}^u$ and a system $\mathcal{MP}$ with m processors of the same computing capacity, if $\mathcal{T}$ is schedulable in $\mathcal{MP}$, the computing capacity of each processor in $\mathcal{MP}$, ξ, must satisfy,

$$\xi \geq \max\{\delta_{\max}(\mathcal{T}^c), 2\lambda_{\max}(\mathcal{T}^u)\} \tag{39}$$

and

$$m \cdot \xi > \Theta(\mathcal{T}^c) + \lambda(\mathcal{T}^u) \tag{40}$$

where

$$\Theta(\mathcal{T}^c) = \max_{t>0}\left(\frac{\sum_{\tau_i^c \in \mathcal{T}^c} DBF(\tau_i^c, t)}{t}\right) \tag{41}$$

Proof: [38] proved $\xi \geq \delta_{max}(\mathcal{T}^c)$ and [27] proved $\xi \geq 2\lambda_{max}(\mathcal{T}^u)$. Thus, we have $\xi \geq \max\{\delta_{\max}(\mathcal{T}^c), 2\lambda_{\max}(\mathcal{T}^u)\}$, which means $\mathcal{T}$ satisfies Eq. (39).

Next, we prove that ξ satisfies Eq. (40). Assuming that $\Theta(\mathcal{T}^c)$ is obtained at $t = t_0$ and consider a sequence of job arrivals over $[0, t_0)$. Since $\mathcal{T}$ is schedulable in $\mathcal{MP}$, we have $\Theta(\mathcal{T}^c)t_0 + \Theta(\mathcal{T}^u)t_0 \leq mt_0\xi$. Note that, $\lambda_i^u = \frac{C_i^u}{V_i} < \frac{C_i^u}{T_i^u}$ for each update transaction τ_i^u. Thus, we have $\lambda(\mathcal{T}^u) = \sum_{\tau_i^u \in \mathcal{T}^u} \lambda_i^u < \Theta(\mathcal{T}^c)$. Since $\Theta(\mathcal{T}^c)t_0 + \Theta(\mathcal{T}^u)t_0 \leq mt_0\xi$, we have $\Theta(\mathcal{T}^c)t_0 + \lambda(\mathcal{T}^u)t_0 < \Theta(\mathcal{T}^c)t_0 + \Theta(\mathcal{T}^u)t_0 < mt_0\xi$, which means $\Theta(\mathcal{T}^c) + \lambda(\mathcal{T}^u) < m\xi$. Based on the above discussions, we have that ξ must satisfy both Eqs. (39) and (40) and the proof is thus finished. □

Based on Lemma 3, we have an important theorem as follows.

Theorem 6: Given a hybrid transaction $\mathcal{T}$ and a system $\mathcal{MP}$ with m unit-capacity processors, $\mathcal{T}$ must be schedulable under EP-HT in $\mathcal{MP}$ if m satisfies

$$m \geq \max\{\frac{2\mathcal{X} - \delta_{max}(\mathcal{T}^c)}{1 - \delta_{max}(\mathcal{T}^c)}, \frac{2\mathcal{X} - 2\lambda_{max}(\mathcal{T}^u)}{1 - 2\lambda_{max}(\mathcal{T}^u)}\} \tag{42}$$

where $\mathcal{X} = \Theta(\mathcal{T}^c) + \lambda(\mathcal{T}^u)$.

Proof: Assuming $\mathcal{T}$ is unschedulable under EP-HT, we prove this theorem by contradiction. Using τ_i to denote the first transaction which cannot be assigned to any processor under EP-HT and using $\overline{\mathcal{T}} = \overline{\mathcal{T}^c} \bigcup \overline{\mathcal{T}^u}$ to denote the hybrid transaction set consisting of the transactions before τ_i, we discuss this theorem in the two cases as follows.

- τ_i is a control transaction. As shown in Algorithm 1, EP-HT always assigns the deadline and the period of each update transaction τ_i^u to $\frac{V_i}{2}$ (at lines 2 to 3) and sorts all transactions by their increasing deadlines (at line 4). Thus, we have that τ_i satisfies Eq. (18) for all transaction sets assigned on processors. Since τ_i cannot be assigned to any processor, we have that $\sum_{\tau_k^c \in \mathcal{T}^c(M_j)} DBF^*(\tau_k^c, D_i^c) + \sum_{\tau_l^c \in \mathcal{T}^u(M_j)} UDBF^*(\tau_k^c, D_i^c) > D_i^c - C_i^c$ for each processor M_j, $1 \leq j \leq m$, based on Theorem 3, which means

$$\sum_{\tau_k^c \in \overline{\mathcal{T}^c}} DBF^*(\tau_k^c, D_i^c) + \sum_{\tau_l^u \in \overline{\mathcal{T}^u}} UDBF^*(\tau_l^u, D_i^c) + C_i^c > m(D_i^c - C_i^c) + C_i^c \tag{43}$$

Since $\Theta(\mathcal{T}^c) \geq \sum_{\tau_i^c \in \mathcal{T}} \frac{DBF(\tau_i^c, D_i^c)}{D_i^c}$ and $\overline{\mathcal{T}} \in \mathcal{T}$, we have

$$\mathcal{X} = \Theta(\mathcal{T}^c) + \lambda(\mathcal{T}^u) \geq \sum_{\tau_k^c \in \overline{\mathcal{T}^c} \bigcup \{\tau_i^c\}} \frac{DBF(\tau_i^c, D_i^c)}{D_i^c} + \sum_{\tau_l^u \in \overline{\mathcal{T}^u}} \frac{C_l^u}{V_j} \tag{44}$$

Note that, all transactions have been sorted in EP-HT, as shown in Algorithm 2. Thus, we can get that τ_i satisfies Eq. (18) for the transaction sets assigned on each processor. So, since $2DBF(\tau_i^c, t) > DBF^*(\tau_i^c, t)$ for each τ_i^c [33] and $2UDBF(\tau_i^c, t) > UDBF^*(\tau_i^c, t)$ (based on Lemma 2), we have

$$2\mathcal{X} \geq 2(\Theta(\overline{\mathcal{T}^c} \bigcup \{\tau_i\}) + \lambda(\overline{\mathcal{T}^u})) \geq \sum_{\tau_k^c \in \overline{\mathcal{T}^c} \bigcup \{\tau_i^c\}} \frac{2DBF(\tau_k^c, D_i^c)}{D_i^c} + 2\sum_{\tau_l^u \in \overline{\mathcal{T}^u}} \frac{C_l^u}{V_l}$$

$$\geq \frac{\sum_{\tau_k^c \in \overline{\mathcal{T}^c}} DBF^*(\tau_k^c, D_i^c) + C_i^c + \sum_{\tau_l^u \in \overline{\mathcal{T}^u}} UDBF^*(\tau_l^u, D_i^c)}{D_i^c}$$
$$\geq m(1 - \frac{C_i^c}{D_i^c}) + \frac{C_i^c}{D_i^c}$$
$$\geq m(1 - \delta_i^c) + \delta_i^c \tag{45}$$

Since $1 - \frac{C_i^c}{D_i^c} \geq 0$, we have
$$m < \frac{2\mathcal{X} - \delta_i^c}{1 - \delta_i^c} \tag{46}$$

Note that, m is a positive integer. So, we have $m \geq 1$ and $2\mathcal{X} \geq 1$ based on Eq. (46). Moreover, let $F(\delta_i^c) = \frac{2\mathcal{X} - \delta_i^c}{1 - \delta_i^c}$, since $2\mathcal{X} \geq 1$ and $0 < \delta_i^c < 1$, we have that $F(\delta_i^c)$ increases with the growth of δ_i^c. Since $\delta_{max}(\mathcal{T}^c) = \max_{1 \leq i \leq n^c}\{\delta_i^c\}$, we have that $F(\delta_i^c) \leq F(\delta_{max}(\mathcal{T}^c))$, which means
$$m < F(\delta_{max}(\mathcal{T}^c)) < \frac{2\mathcal{X} - \delta_{max}(\mathcal{T}^c)}{1 - \delta_{max}(\mathcal{T}^c)} \tag{47}$$

Clearly, Eqs. (42) and (47) contradict each other, which means that τ_i must be schedulable under EP-HT.

- τ_i is an update transaction. The proof process is similar to the first case. We can easily get
$$m < \frac{2\mathcal{X} - 2\lambda_i^u}{1 - 2\lambda_i^u} \tag{48}$$

Note that, m is a positive integer. So, we have $m \geq 1$, which means $2\mathcal{X} \geq 1$ based on Eq. (48). Moreover, let $F(\lambda_i^u) = \frac{2\mathcal{X} - 2\lambda_i^u}{1 - 2\lambda_i^u}$, since $2\mathcal{X} \geq 1$ and $0 < 2\lambda_i^u < 1$, we have that $F(\lambda_i^u)$ increases with the growth of λ_i^u. Since $\lambda_{max}(\mathcal{T}^u) = \max_{1 \leq i \leq n^u}\{\lambda_i^u\}$, we have that $F(\lambda_i^u) \leq F(\lambda_{max}(\mathcal{T}^u))$, which means
$$m < F(\lambda_{max}(\mathcal{T}^u)) < \frac{2\mathcal{X} - 2\lambda_{max}(\mathcal{T}^u)}{1 - 2\lambda_{max}(\mathcal{T}^u)} \tag{49}$$

Clearly, Eqs. (42) and (49) contradict each other, which means that τ_i must be schedulable under EP-HT.

As discussed above, we have that $\mathcal{T}$ must be schedulable under EP-HT if m satisfies Eq. (42) and the proof is thus finished. □

Based on Theorem 6, we now present a resource augmentation result regarding EP-HT.

Theorem 7: Algorithm **EP-HT** *can guarantee the following performance: if a hybrid transaction set $\mathcal{T} = \mathcal{T}^c \bigcup \mathcal{T}^u$ is schedulable on m identical processor each of computing capacity ξ, then $\mathcal{T}$ must be schedulable under* **EP-HT** *on m processors that are each $(3 - \frac{1}{m})$ times as far as the original.*

Proof: Note that, $\mathcal{T}$ is schedulable on m ξ-speed processors. So, based on Lemma 3, we have $\delta_{\max}^c \leq \xi$, $\lambda_{\max} \leq \frac{1}{2}\xi$ and $\mathcal{X} < m\xi$, where $\mathcal{X} = \Theta(\mathcal{T}^c) + \lambda(\mathcal{T}^u)$. Afterwards, based on Eq. (42), we have
$$m \geq \max\{\frac{2\mathcal{X} - \delta_{\max}(\mathcal{T}^c)}{1 - \delta_{\max}(\mathcal{T}^c)}, \frac{2\mathcal{X} - 2\lambda_{\max}(\mathcal{T}^u)}{1 - 2\lambda_{\max}(\mathcal{T}^u)}\}$$
$$\Leftarrow m \geq \frac{2m\xi - \xi}{1 - \xi}$$
$$\equiv \xi \leq \frac{m}{3m - 1}$$
which is as claimed in the theorem. □

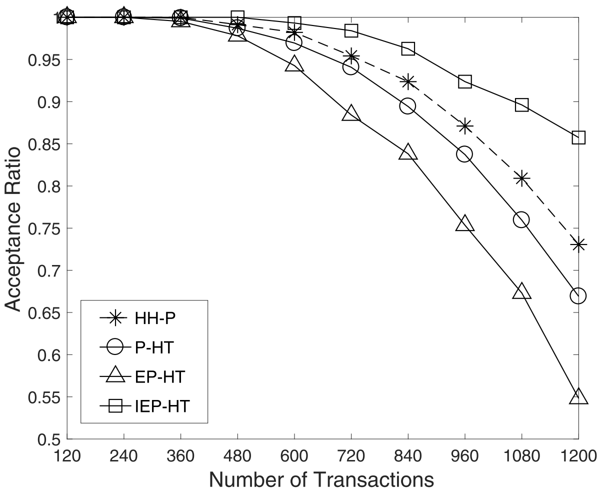


FIGURE 1. **Acceptance ratio comparison.**

VI. SIMULATION RESULTS

In this section, we evaluate the performances of HH-P, P-HT, EP-HT and IEP-HT in terms of acceptance ratio, average execution time and processor workload. Clearly, a favorable scheduling method means a higher acceptance ratio, a shorter average execution time and a lighter processor workload.

A. EXPERIMENTAL SETTING

In our experiments, we test the performances of the three proposed methods in a system with 4 processors. Each hybrid transaction set $\mathcal{T}$ in the simulation consists of $\lfloor 0.8N \rfloor$ update transactions and $N - \lfloor 0.8N \rfloor$ control transactions, where N is the number of the transactions in $\mathcal{T}$. The WCET of each update transaction is uniformly distributed over [1, 15] and the validity interval length of each real-time data object is uniformly distributed over [20, 16000]. The WCET, the deadline and the period of each control transaction are uniformly distributed over [1, 15], [300, 1200] and [600, 2400], respectively. 10000 hybrid transaction sets are generated randomly to test the performances of the proposed three methods.

B. EXPERIMENTAL RESULTS

The performances of the intuitive method and the three proposed methods are tested under different N. In our experiments, we always initialize N to 120 and add it for 120 in each step, until it reaches 1200. Fig. 1, 2 and 3 show the performances of the proposed methods.

1) COMPARISON OF ACCEPTANCE RATIOS

As shown in Fig. 1, the acceptance ratios of all methods decrease with the growth of N. IEP-HT always has a better performance than the others. Moreover, the advantage of them are more obvious under a larger N. At $N = 1200$, IEP-HT can improve the acceptance ratios of HH-P, P-HT and EP-HT for about 17%, 28% and 56%, respectively. The reason as follows: 1) IEP-HT can make the hybrid transaction set schedulable by assigning a lower deadline to the update

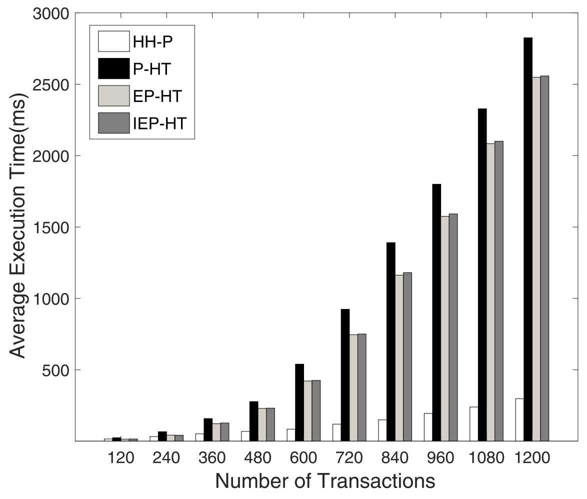


FIGURE 2. Average execution time comparison.

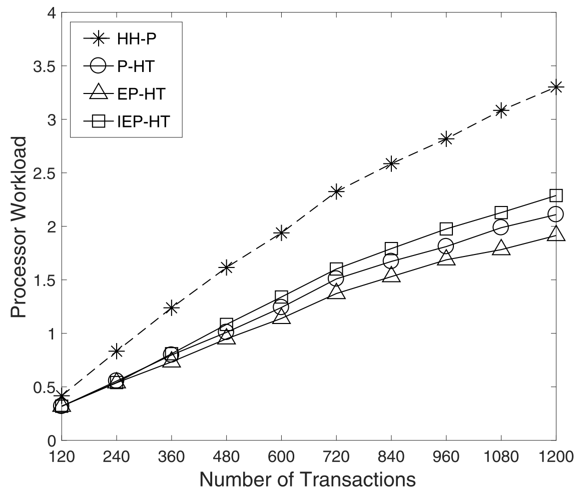


FIGURE 3. Processor workload comparison.

transaction compared with HH-P, P-HT, and EP-HT; 2) the sufficient condition in EP-HT is more stringent than those of the other three methods, EP-HT has the worst performance on acceptance ratio; 3) because of $DBF(\tau_i^u, t) \leq UDBF(\tau_i^u, t)$ is established, the acceptance ratio of P-HT is lower than that of HH-P.

2) COMPARISON OF AVERAGE EXECUTION TIMES

Fig. 2 shows that the average execution times of all methods increase with the growth of N. HH-P always has the best performance in time consumption. It is because that MDC is executed in P-HT, EP-HT and IEP-HT, while HH-P assigns the deadline and the period of each update transaction τ_i to $\frac{V_i}{2}$ directly. In addition, the gap between the average execution times (of the four methods) increases with the increasing of N and the average execution time of HH-P is only about 10%, 12% and 12% of those of P-HT, EP-HT and IEP-HT, respectively, at $N = 1200$. Note that, EP-HT has almost the same time performance as IEP-HT. This is because that both algorithms can calculate the DBF approximately.

3) COMPARISON OF PROCESSOR WORKLOADS

As depicted in Fig. 3, EP-HT always has a better performance than the other methods and the performances of P-HT, EP-HT and IEP-HT are always better than that of HH-P. At the largest gap, the processor workload under EP-HT is only about 58%, 91% and 84% of those under HH-P, P-HT and IEP-HT, respectively. The reason as follows: 1) the periods of update transactions under P-HT, EP-HT and IEP-HT are always larger than those under HH-P, HH-P has the worst workload performance; 2) since the conditions in Theorems 3 and 4 are more strict than those in Theorem 2, EP-HT assigns transactions to processors more averagely than P-HT; 3) because MDC is utilized after all transactions are successfully dispatched in EP-HT, it assigns transactions to processors more averagely than IEP-HT.

VII. CONCLUSIONS AND FUTURE WORK

In this paper, we studied the co-scheduling of hybrid transactions on multiprocessor platforms. To the best of our knowledge, this is the first attempt to propose the multiprocessor co-scheduling methods of hybrid transactions. Two intuitive partitioned scheduling methods (for hybrid transaction sets in multiprocessor systems) were proposed by developing some existing scheduling methods to multiprocessor environment. For improving the efficiency of these two methods, we proposed two improvement strategies which reduce the time for testing the schedulability of transaction sets. Theoretical analysis proves that the resource argumentations of the two improvement strategies can reach $3 - \frac{1}{m}$. A set of experiments is conducted to evaluate the performs of the proposed methods in terms of acceptance ratio, average execution time and workload of processors.

For future work, we will extend the proposed methods to deal with jitter-based systems. Moreover, the global multiprocessor scheduling of hybrid transactions is another issue that we plan to study.

REFERENCES

[1] M. Canizo, A. Conde, S. Charramendieta, R. Miñón, R. Cid-Fuentes, and E. C. Onieva, "Implementation of a large-scale platform for cyber-physical system real-time monitoring," *IEEE Access*, vol. 7, pp. 52455–52466, 2019.
[2] T. Gustafsson and J. Hansson, "Data management in real-time systems: A case of on-demand updates in vehicle control systems," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp.,* Toronto, Canada, May 2004, pp. 182–191.
[3] X. Shi, Y. Shen, Y. Wang, and L. Bai, "Differential-clustering compression algorithm for real-time aerospace telemetry data," *IEEE Access*, vol. 6, pp. 57425–57433, 2018.
[4] J. Ko, C. Lu, M. B. Srivastava, J. A. Stankovic, A. Terzis, and M. Welsh, "Wireless sensor networks for healthcare," *Proc. IEEE*, vol. 98, no. 11, pp. 1947–1960, Nov. 2010.
[5] S. Han, A. K. Mok, J. Meng, Y. H. Wei, P. C. Huang, Q. Leng, X. Zhu, L. Sentis, K. S. Kim, and R. Miikkulainen, "Architecture of a cyberphysical avatar," in *Proc. ACM/IEEE Int. Conf. Cyber-Phys. Syst.,* Philadelphia, PA, USA, Apr. 2013, pp. 189–198.
[6] 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, 1973.
[7] J. Y.-T. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic, real-time tasks," *Perform. Eval.*, vol. 2, no. 4, pp. 237–250, Dec. 1982.

[8] S. Han, K.-Y. Lam, J. Wang, K. Ramamritham, and A. K. Mok, ''On co-scheduling of update and control transactions in real-time sensing and control systems: Algorithms, analysis, and performance,'' *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2325–2342, Oct. 2013.

[9] K. Ramamritham, ''Real-time databases,'' *Distrib. Parallel Databases*, vol. 1, no. 2, pp. 199–226, Apr. 1993.

[10] S. J. Ho, T. W. Kuo, and A. K. Mok, ''Similarity-based load adjustment for real-time data-intensive applications,'' in *Proc. IEEE Real-Time Syst. Symp.,* San Francisco, CA, USA, Dec. 1997, pp. 144–157.

[11] M. Xiong, S. Han, and K.-Y. Lam, ''A deferrable scheduling algorithm for real-time transactions maintaining data freshness,'' in *Proc. IEEE Real-Time Syst. Symp.,* Miami, FL, USA, Dec. 2005, pp. 27–37.

[12] M. Xiong and K. Ramamritham, ''Deriving deadlines and periods for real-time update transactions,'' *IEEE Trans. Comput.*, vol. 53, no. 5, pp. 567–583, May 2004.

[13] Y. Kim and S. Son, ''Predictability and consistency in real-time database systems,'' *IEEE Real-Time Syst. Newslett.*, vol. 5, nos. 2–3, pp. 42–45, Mar. 1993.

[14] Y. Jin, J. P. Hamiez, and J. K. Hao, ''Algorithms for the minimum sum coloring problem: A review,'' *Artif. Intell. Rev.*, vol. 47, no. 3, pp. 367–394, Mar. 2017.

[15] Y. Jin and J.-K. Hao, ''Hybrid evolutionary search for the minimum sum coloring problem of graphs,'' *Inf. Sci.*, vols. 352–353, pp. 15–34, Jul. 2016.

[16] K.-D. Kang, S. H. Son, J. A. Stankovic, and T. F. Abdelzaher, ''A QoS-sensitive approach for timeliness and freshness guarantees in real-time databases,'' in *Proc. EuroMicro Conf. Real-Time Syst.,* Vienna, Austria, Jun. 2002, pp. 203–212.

[17] K.-Y. Lam, M. Xiong, B. Liang, and Y. Guo, ''Statistical quality of service guarantee for temporal consistency of real-time data objects,'' in *Proc. IEEE Real-Time Syst. Symp.*, Lisbon, Portugal, Dec. 2004, pp. 276–285.

[18] X. Song and J. W. S. Liu, ''Maintaining temporal consistency: Pessimistic vs. Optimistic concurrency control,'' *IEEE Trans. Knowl. Data Eng.*, vol. 7, no. 5, pp. 786–796, Oct. 1995.

[19] T.-W. Kuo and A. K. Mok, ''SSP: A semantics-based protocol for real-time data access,'' in *Proc. Real-Time Syst. Symp.*, Dec. 1993, pp. 76–86.

[20] M. Xiong, S. Han, K.-Y. Lam, and D. Chen, ''Deferrable scheduling for maintaining real-time data freshness: Algorithms, analysis, and results,'' *IEEE Trans. Comput.*, vol. 57, no. 7, pp. 952–964, Jul. 2008.

[21] S. Han, D. Chen, M. Xiong, K.-Y. Lam, A. K. Mok, and K. Ramamritham, ''Schedulability analysis of deferrablescheduling algorithms for maintainingreal-time data freshness,'' *IEEE Trans. Knowl. Data Eng.*, vol. 63, no. 4, pp. 979–994, Apr. 2014.

[22] A. K. Jha, M. Xiong, and K. Ramamritham, ''Mutual consistency in real-time databases,'' in *Proc. IEEE Real-Time Syst. Symp.,* Dec. 2006, pp. 335–343.

[23] S. Han, D. Chen, M. Xiong, and A. K. Mok, ''Online scheduling switch for maintaining data freshness in flexible real-time systems,'' in *Proc. IEEE Real-Time Syst. Symp.,* Washington, DC, USA, Dec. 2009, pp. 115–124.

[24] S. Han, K.-Y. Lam, J. Wang, S. H. Son, and A. K. Mok, ''Adaptive co-scheduling for periodic application and update transactions in real-time database systems,'' *J. Syst. Softw.*, vol. 85, no. 8, pp. 1729–1743, Aug. 2012.

[25] J. Wang, K.-Y. Lam, S. Han, S. H. Son, and A. K. Mok, ''An effective fixed priority co-scheduling algorithm for periodic update and application transactions,'' *Computing*, vol. 95, nos. 10–11, pp. 993–1018, Oct. 2013.

[26] L. Lundberg, ''Multiprocessor scheduling of age constraint processes,'' in *Proc. IEEE Int. Conf. Embedded Real-Time Comput. Syst. Appl.,* Hiroshima, Japan, Oct. 1998, p. 42.

[27] J. Li, J.-J. Chen, M. Xiong, and G. Li, ''Workload-aware partitioning for maintaining temporal consistency upon multiprocessor platforms,'' in *Proc. IEEE Real-Time Syst. Symp.,* Vienna, Austria, Nov. 2011, pp. 126–135.

[28] J. Li, J.-J. Chen, M. Xiong, G. Li, and W. Wei, ''Temporal consistency maintenance upon partitioned multiprocessor platforms,'' *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1632–1645, May 2016.

[29] J. Wang, S. Han, K.-Y. Lam, and A. K. Mok, ''Maintaining data temporal consistency in distributed real-time systems,'' *Real-Time Syst.*, vol. 48, no. 4, pp. 387–429, Jul. 2012.

[30] H. Hoang, G. Buttazzo, M. Jonsson, and S. Karlsson, ''Computing the minimum EDF feasible deadline in periodic systems,'' in *Proc. IEEE Embedded Real-Time Comput. Syst. Appl.,* Hakodate, Japan, Aug. 2006, pp. 125–134.

[31] F. Zhang and A. Burns, ''Schedulability analysis for real-time systems with EDF scheduling,'' *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1250–1258, Sep. 2009.

[32] G. Li, C. Deng, J. Li, Q. Zhou, and W. Wei, ''Deadline and period assignment for update transactions in co-scheduling environment,'' *IEEE Trans. Comput.*, vol. 66, no. 7, pp. 1119–1131, Jul. 2017.

[33] K. Albers and F. Slomka, ''An event stream driven approximation for the analysis of real-time systems,'' in *Proc. EuroMicro Conf. Real-Time Syst.,* Catania, Italy, Jun. 2004, pp. 187–195.

[34] S. Baruah and N. Fisher, ''The partitioned multiprocessor scheduling of sporadic task systems,'' in *Proc. IEEE Real-Time Syst. Symp.,* Miami, FL, USA, Dec. 2005, pp. 321–329.

[35] J.-J. Chen and S. Chakraborty, ''Resource augmentation bounds for approximate demand bound functions,'' in *Proc. IEEE Real-Time Syst. Symp.,* Vienna, Austria, Nov. 2011, pp. 272–281.

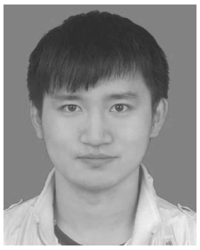
[36] N. Fisher, S. Baruah, and T. P. Baker, ''The partitioned scheduling of sporadic tasks according to static-priorities,'' in *Proc. EuroMicro Conf. Real-Time Syst.,* Dresden, Germany, Jul. 2006, pp. 118–127.

[37] C. A. Phillips, C. Stein, E. Torng, and J. Wein, ''Optimal time-critical scheduling via resource augmentation,'' in *Proc. ACM Symp. Theory Comput.*, El Paso, TX, USA, May 1997, pp. 140–149.

[38] S. Baruah and N. Fisher, ''The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems,'' *IEEE Trans. Comput.*, vol. 55, no. 7, pp. 918–923, Jul. 2006.

CHENGGANG DENG received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2017, where he is currently a Postdoctoral with the School of Computer Science and Technology. His research interests include real-time database and real-time scheduling.

GUOHUI LI received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 1999, where he is currently the Dean with the School of Software Engineering. He was the Full Professor, in 2004. His research interests mainly include real-time systems, mobile computing, and advanced data management.



QUAN ZHOU received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2015. He was a Postdoctoral with the School of Computer Science and Technology, HUST, where he is currently an Assistant Professor. He serves as the Corresponding Author of this paper. His research interests include real-time scheduling and mobile computing.

JIANJUN LI received the Ph.D. degree in computer science from the Huazhong University of Science and Technology (HUST), Wuhan, China, in 2012, where he is currently an Associate Professor with the School of Computer Science and Technology. He was a Senior Research Associate with the Department of Computer Science, City University of Hong Kong, Kowloon, Hong Kong. His research interests include real-time systems and advanced data management.