An Enhanced Real-Time Deferrable Server Scheduler for Xen Virtualization Systems

Jun Wu and Jian-Fu Li

***Abstract*—Real-time deferrable server (RTDS) scheduler is an experimental CPU scheduler for Xen virtualization systems since version 4.5. Under RTDS, each virtual CPU (VCPU) is guaranteed to have a predefined amount of physical CPU (PCPU) capacity so that the performance can be better predicted. However, the guaranteed capacity might not fit the requirement of a VCPU at the run-time because it is defined offline. Therefore, the performance of virtual machines (VMs) might be deteriorated at the run-time. In this paper, an RTDS-based CPU scheduler is proposed, called enhanced real-time deferrable server (ERTDS), to provide an additional amount of PCPU capacity to a VCPU when its run-time requirement is higher than expected. We have implemented ERTDS in Xen version 4.7 and a series of experiments has been conducted for which we have some encouraging results.**

***Index Terms*—Xen virtualization system, CPU Scheduler, real-time deferrable server**

I. Introduction

XEN was originally proposed by Keir Fraser [1] as a research project at the University of Cambridge. It enables multiple virtual machines (VMs) to be running on a single physical machine (PM) isolatedly. Later, Xen has been released as an open source project [2], and it has become a leading virtualization platform. Based on Xen, many researchers have explored the use of virtualization technology for various application domains, such as fault tolerance [3], encryption [4], simulation [5], and cloud computing [6]. When different applications are considered, the VMs of a Xen virtualized system may have different resource requirements. Therefore, the allocation of limited underlying physical resources, such as CPU and memory, has become an active research topic. In this paper, we are interested in scheduling of physical CPUs (PCPUs) to virtual CPUs (VCPUs) since it is the major performance-dominated resource for VMs.

In the past decade, Xen has released several CPU schedulers to schedule the virtual CPUs (VCPUs) of VMs on the underlying PCPUs of the PM, such as *Credit* [7], *simple earliest deadline first* (SEDF) [8], *borrowed virtual time* (BVT) [9], and *real-time deferrable server* (RTDS) [10]. Note that SEDF and BVT were not supported since Xen 4.6 and 3.0, respectively. At the current stage, Credit and RTDS are two major CPU schedulers but with different design philosophies: Credit is a proportional fairness scheduler while RTDS is a real-time CPU scheduler built to provide a guaranteed PCPU capacity to every VCPU on SMP or multi-core hosts. Since Credit (and its successor Credit2 scheduler [11]) is the default scheduler of Xen, many excellent work has been proposed for improving the performance of Credit (such as [12], [13], [14], [15], [16], [17], [18], [19], [20]), however, little work has been done for RTDS. To the best of our knowledge, only very few work has been addressed for RTDS. In particular, Xi *et al.* [21] have proposed an RTDS-based CPU resource management approach for real-time cloud computing.

RTDS uses a static resource allocation strategy to allocate the underlying PCPU to VCPUs, i.e., in an off-line fashion. In particular, the PCPU allocation (i.e., the guaranteed PCPU capacity) of each VCPU is defined off-line by two parameters: *budget* and *period*[1]. According to the pre-defined parameters, RTDS guarantees that every VCPU can be running on a PCPU for up to the time defined by its budget for every period of time. In other words, RTDS provides a guaranteed and predictable PCPU capacity to each VCPU such that it is capable to support VCPUs with real-time workloads. Nowadays, many applications that run on a Xen virtualized system are with different criticalities. Such a system, called *Xen virtualized mixed-criticality system* [22], consists of RT-VMs and NRT-VMs which are VMs with real-time and non-real-time workloads, respectively.

Since the timing requirements of RT-VMs are known a prior, their required PCPU capacities can be guaranteed by RTDS in terms of well parameter settings. However, the performance of NRT-VMs cannot be guaranteed because the on-line requirements of non-real-time workloads are vary and unpredictable. As the results, the performance of a NRT-VM might be deteriorated even if RTDS provides a guaranteed PCPU capacity (which might not fit the run-time requirements of the NRT-VM). To improve the performance of VMs, the underlying physical resources must be allocated wisely and to be adjusted dynamically at the run-time so that the on-line requirements can be met.

In this paper, an RTDS-based CPU scheduler, called *enhanced real-time deferrable server* (ERTDS) scheduler, is proposed to solve such a performance deteriorated problem. Based on the schedulability analysis of RTDS, ERTDS assigns a proper amount of PCPU capacity to every VCPU (note that such an amount is still guaranteed at the run-time) and creates a fake VCPU with a preserved amount of PCPU capacity. The fake VCPU will provide additional PCPU capacity to a VCPU at the run-time when its run-

Manuscript received January 15, 2018; revised June 1, 2018. This work was supported in part by the Ministry of Science and Technology (MOST) of Taiwan under grants MOST-105-2628-E-153 -001-MY2 and MOST-104-2815-C-153-004-E.

Jun Wu and Jian-Fu Li are with the Department of Computer Science and Information Engineering, National Pingtung University, 900 Pingtung City, Taiwan, R.O.C. e-mail: junwu@mail.nptu.edu.tw

An earlier version of this paper, entitled "ERTDS: A Dynamic CPU Scheduler for Xen Virtualization Systems", was presented at the 2017 IEEE International Conference on Applied System Innovation (ICASI). The results have been extended in exploring the schedulability and the analysis of capacity guarantee for VCPUs. More examples, figures, experimental results, and proofs are included in this extension.

[1]Note that the parameters are allowed to be adjusted manually at the run-time.

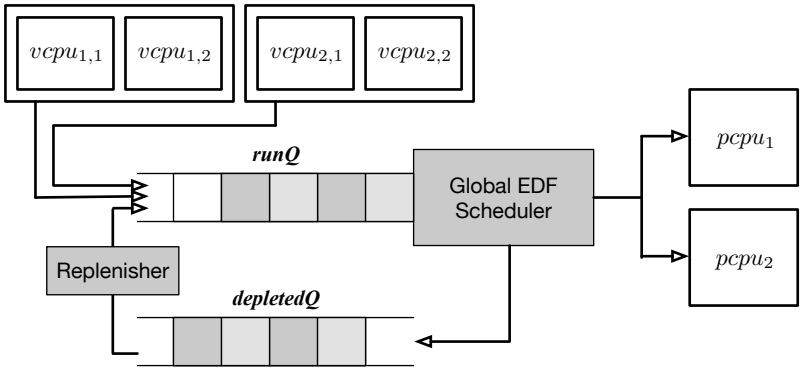

Fig. 1. Real-Time deferrable server (RTDS).

time requirement is higher than expected. Therefore, when ERTDS is adopted, the performance deteriorated problem can be solved while still provide a guaranteed amount of PCPU capacity to each VM. Note that ERTDS was first proposed in our previous work [23]. In this paper, the results have been extended in exploring the schedulability and the analysis of capacity guarantee for VCPUs. Furthermore, more examples, figures, experimental results, and proofs are included in this paper.

The major contributions of this research are two-fold: (1) We explore the schedulability and the properties of ERTDS scheduler. Therefore, a proper amount of guaranteed PCPU capacity can be determined for each VCPU off-line such that the utilization of the PCPU and the performance of RT-VMs can be better predicted; and (2) Our proposed ERTDS scheduler provides an additional PCPU capacity to each VCPU when its run-time requirements is higher than expected for which the performance of NRT-VMs could be improved greatly. Note that our proposed ERTDS scheduler has been implemented in Xen 4.7 and a series of experiments has been conducted for which some encourage results were obtained. In particular, the experimental results show that ERTDS outperforms the original RTDS.

The rest of this paper is organized as follows: Section II presents the preliminary and the motivations of this research. Section III proposes ERTDS and provides an example to illustrate the details. Section IV provides the properties and the schedulability analysis of our proposed ERTDS. Section V reports the performance evaluation. Section VI is the conclusion and the future work.

II. Preliminary and Motivations

In this section, the preliminary of this research and the system model are presented. We also discuss a potential performance deterioration problem of the original RTDS scheduler which motivates this research.

A. System Model

We consider a Xen virtualization system which consists of a set of $n+1$ virtual machines $\mathbf{VM} = \{vm_0, vm_1, \cdots, vm_n\}$, where vm_0 is initially started by Xen at the boot time with a higher privilege for the management purposes (such as to create or to terminate other VMs, to control the scheduling behaviors, and to allocate the physical resources). Although there are several types of physical resources, we are only interested in the physical CPUs (PCPUs) as we mentioned earlier. We assume that there exists a set of m PCPUs in the system, denoted as $\mathbf{PCPU} = \{pcpu_1, pcpu_2, \cdots, pcpu_m\}$. Note that each $pcup_j$ represents a processing core in the physical machine (PM). Each VM vm_i has n_i virtual CPUs (VCPUs) denoted by $\mathbf{VCPU}_i = \{vcpu_{i,1}, vcpu_{i,2}, \cdots, vcpu_{i,n_i}\}$. We also use $\mathbf{VCPU}$ to denote the set of all VCPUs in the system, i.e., $\mathbf{VCPU} = \bigcup_{1 \le i \le n} \mathbf{VCPU}_i$.

B. The original RTDS scheduler

Real-time deferrable server (RTDS) scheduler provides a guaranteed PCPU capacity to each VCPU of each VM in a static manner. In particular, RTDS scheduler models a VCPU as a *deferrable server* [24]. It assigns two parameters *period* and *budget* to each VCPU statically. Let period and budget of a $vcpu_{i,j}$ denote as $P_{i,j}$ and $B_{i,j}$. Under RTDS, it is guaranteed that every $vcpu_{i,j}$ can be running on a PCPU for up to $B_{i,j}$ μs (not necessarily continuously) for every $P_{i,j}$ μs. In other words, RTDS guarantees a $B_{i,j}/P_{i,j}$ PCPU utilization to every $vcpu_{i,j}$ at the run-time.

As shown in Figure 1, RTDS uses two global queues $runQ$ and $depletedQ$ to manage the PCPU scheduling of VCPUs with and without budget. Note that we use $b_{i,j}$ to denote the current value of the budget of a $vcpu_{i,j}$. In particular, the global $runQ$ is an ordered priority queue holds all VCPUs which have a positive value of budget (i.e., $b_{i,j} > 0$, $\forall vcpu_{i,j} \in runQ$) and sorted by VCPU's deadlines. Note that the deadline of a VCPU is at the end of its current period. The $depletedQ$ holds all VCPUs whose budget is depleted (i.e., $b_{i,j} = 0$, $\forall vcpu_{i,j} \in depletedQ$) and it is a unordered queue. The budget exhaustion and replenishment of RTDS as follows:

Initially, the value of each $vcpu_{i,j}$'s budget is set as $B_{i,j}$, i.e., $b_{i,j} = B_{i,j}$. Whenever a $vcpu_{i,j}$ becomes runnable (i.e., there are jobs required to be executed on $vcpu_{i,j}$, or more simply, $vcpu_{i,j}$ is not idle.), it will be added into the global $runQ$, while it will be removed from the $runQ$ when it becomes idle. All VCPUs in the $runQ$ are scheduled by the *preemptive global earliest deadline first* (G-EDF) algorithm [25]. It always selects the highest priority VCPU from the $runQ$ to run on a feasible PCPU. Note that the highest priority VCPU is the one with the earliest deadline among all VCPUs in the $runQ$. Also note that a PCPU is feasible if it is idle or it has a lower priority VCPU running on it (i.e.,

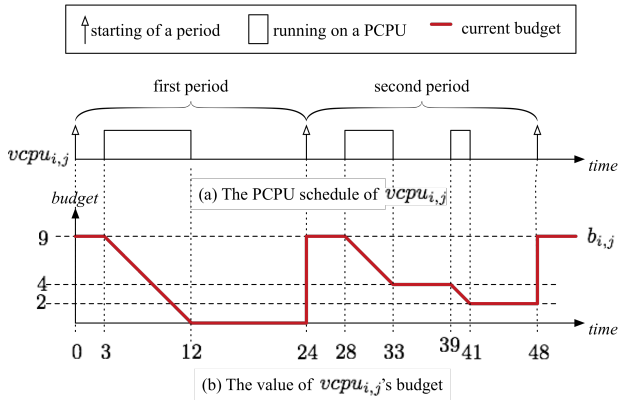

Fig. 2. An example schedule of $vcpu_{i,j}$ and its budget $b_{i,j}$.

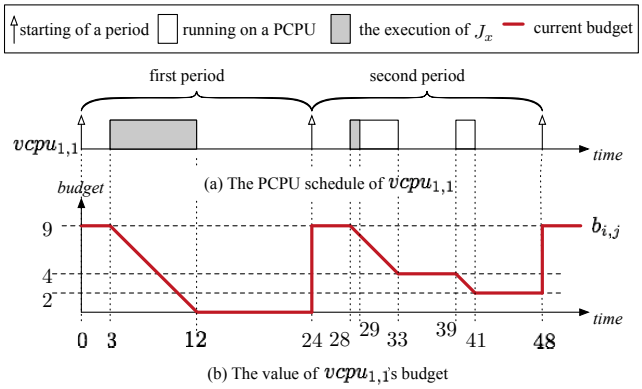

Fig. 3. An example schedule of $vcpu_{i,j}$ with a job J_x, where the budget of $vcpu_{i,j}$ is 9 μs and the computation time of J_x is 10 μs.

a running VCPU with a longer deadline). When a $vcpu_{i,j}$ is running on a PCPU, its budget is continuously burned. Note that the budget of a VCPU is preserved within each of its periods. In other words, it will be guaranteed that a VCPU $vcpu_{i,j}$ can be running on a PCPU up to $B_{i,j}$ μs for every $P_{i,j}$ μs.

Also note that a VCPU's budget will be preserved within each period. In particular, the remaining budget of a VCPU is preserved until its deadline (i.e., the beginning of its next period) when it changes from activated status to idle. However, the remaining budget (if any) will be discarded at the end of each period. On the other hand, a VCPU will be moved from the $runQ$ to the $depletedQ$ if its budget is depleted earlier than its deadline. More specifically, it cannot be run on any PCPU until its next period. Note that a VCPU has its budget replenished at the beginning of each of its periods. The following lemma shows the major feature of RTDS:

Lemma 1: RTDS provide a guaranteed PCPU capacity $\frac{B_{i,j}}{P_{i,j}}$ for each VCPU $vcpu_i$, i.e., each $vcpu_{i,j}$ is guaranteed to be running on a feasible PCPU up to $B_{i,j}$ μs for every $P_{i,j}$ μs.

We use the following example to illustrate the details of RTDS:

Example 1: Consider a VCPU which has a period $P_{i,j} = 24\mu$s and a budget $B_{i,j} = 9\mu$s. In other words, $vcpu_{i,j}$ has a guaranteed 9/24=37.5% PCPU utilization, i.e., it can be running on a PCPU up to 9 μs for every 24 μs. Figure 2 shows an example PCPU schedule and its corresponding budget for $vcpu_{i,j}$'s first two periods. As shown in Figure 2, at time 0, $vcpu_{i,j}$ starts its first period, later at time 3, it becomes the highest priority VCPU in the $runQ$ (i.e., it has an earliest deadline among all VCPUs in the $runQ$) and starts its execution on an allocated PCPU and burns its budget continuously. Note that we use white boxes to represent the running of VCPUs.

Since the budget of $vcpu_{i,j}$ is 9 μs, $vcpu_{i,j}$ is only allowed to be running on a PCPU up to 9 μs. As the result, $vcpu_{i,j}$ continuously runs and burns its budget until its $b_{i,j} = 0$, i.e., at time 12. Note that at time 12, $vcpu_{i,j}$ must stop running and has been moved to the $depletedQ$ since its budget is already depleted. At time 24, $vcpu_{i,j}$ starts its next period and its budget has been replenished, i.e., $b_{i,j} = B_{i,j} = 9$. Also note that, at time 24, $vcpu_{i,j}$ has been moved to the $runQ$ since its $b_{i,j} > 0$. Later, at time 28, $vcpu_{i,j}$ becomes the highest priority VCPU in the $runQ$, it starts its execution and its budget also starts to burn. Suppose that at time 33, $vcpu_{i,j}$ becomes idle or has been preempted by other higher priority VCPUs. Note that the remaining budget is preserved for its further execution before its deadline (i.e., before its next period). Later at time 39, $vcpu_{i,j}$ resumes its execution and burns its remaining budget. Suppose that at time 41, $vcpu_{i,j}$ becomes idle or has been preempted again. Note that its remaining budget will be discarded at the end of its deadline. Also note that it will be replenished to $B_{i,j}$ at the beginning of its next period. □

C. Motivations

The following example shows a potential performance issue:

Example 2: Consider the example schedule shown in Figure 2 again. Suppose that there is a job with 10 μs computation time has arrived at time 3 on $vcpu_{i,j}$ and it starts its execution immediately since $vcpu_{i,j}$ is just allocated to a PCPU (note that we assume the job J_x is the highest priority job on $vcpu_{i,j}$). Figure 3 shows the schedule and the corresponding budget for $vcpu_{1,1}$. As shown in the figure, the job J_x cannot complete within $vcpu_{i,j}$'s first period since the budget of $vcpu_{i,j}$ is 9 μs. Thus, at time 12, the job J_x has been suspended. Later, at time 28, $vcpu_{i,j}$ resumes its execution and J_x's remaining computation is completed at time 29. In this case, the response time of the job is 29-3 = 26 μs. Now, suppose we can increase the budget of $vcpu_{i,j}$ to 10 μs, as shown in Figure 4, the job J_x can be completed at time 13, thus, its response time is reduced to 10 μs. Compare to 26μs, it is a great improvement. □

The major purpose of RTDS scheduler is to provide a predefined amount of PCPU capacity (i.e., utilization) to each VCPU, i.e., $vcpu_{i,j}$ is guaranteed to have a 9/24=37.5% PCPU capacity in Example 1 for which it can be running on a PCPU up to 9 μs for every 24 μs. However, Example 2 shows the predefined budget of $vcpu_{i,j}$ might not fit its on-line requirement since the on-line workload is not predictable. When the on-line requirement of a VCPU is higher than

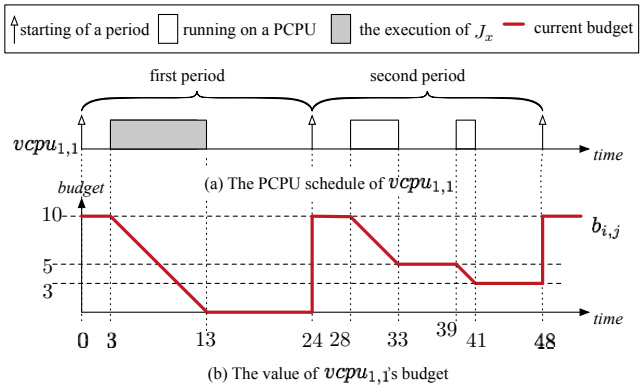

Fig. 4. An example schedule of $vcpu_{i,j}$ with a job J_x, where the budget of $vcpu_{i,j}$ is 10 μs and the computation time of J_x is 10 μs.

its predefined budget, the performance of the jobs might be deteriorated. We observed in Example 2 that a minor increasing of VCPU's budget could result in a great performance improvement (as shown in Figure 4). Such an observation motivates this research. We shall present our approach in the next section.

III. ERTDS: Enhanced Real-Time Deferrable Server Scheduler

In this section, we shall propose a CPU scheduler for Xen to solve the potential performance deterioration problem mentioned in Example 2. The proposed scheduler is called *enhanced real-time deferrable server* (ERTDS) which is a RTDS-based scheduler. The main purposes of ERTDS are two-fold:

1) To increase the predictability of the performance of each VCPU, ERTDS provides a guaranteed PCPU capacity to each VCPU;
2) To improve the responsiveness of each VCPU, ERTDS provides additional PCPU capacity to a VCPU when it has depleted its budget.

We now present ERTDS as follows: our proposed ERTDS scheduler is the same as RTDS scheduler except it can provide additional budget to VCPUs. In particular, ERTDS creates a *fake VCPU* $vcpu^*$ with a period P^* and a budget B^*, respectively. For convenient, we also denote the current value of its budget as b^*. The period of $vcpu^*$ is set as the *least common multiple* (LCM) of all VCPUs' periods. In other words, its period equals to the *hyperperiod* of all VCPUs. Note that the budget of $vcpu^*$ can be treated as a *global budget* which can be provided to other VCPUs dynamically at the run-time. As we mentioned earlier, all VCPUs in the $runQ$ are scheduled by the G-EDF algorithm. In other words, a VCPU with the earliest deadline among all others will be selected to be running on a feasible PCPU. Since the period of the fake VCPU is equal to the hyperperiod of all VCPUs, it will has the lowest priority among all VCPU when G-EDF is adopted. Therefore, the fake VCPU $vcpu^*$ will be scheduled to be running on a feasible PCPU only if it is the only one VCPU in the $runQ$.

Whenever ERTDS selects the fake VCPU, it will not be running on a feasible PCPU. Instead, ERTDS will select a VCPU from the $depletedQ$ to be running on a feasible PCPU. Since a VCPU selected from the $depletedQ$ has depleted its budget, it will be running by using the additional budget, i.e., the global budget, from the fake VCPU. Such an additional budget provides the VCPU a chance to be running earlier (i.e., before the beginning of its next period). Also note that the priority of the selected VCPU is set as the priority of the fake VCPU (i.e., the lowest priority) when it is running with the additional budget. It allows another VCPU to preempt the execution of the selected VCPU when it budget has been replenished or it becomes activated from idle.

Since ERTDS is the same as RTDS except the additional budget supplement, ERTDS not only provides a guaranteed PCPU capacity to each VCPU, but at the same time, it also provides an additional budget to the VCPUs which have been depleted their budgets. As the result, the performance of the selected VCPUs can be improved. When the $depletedQ$ has more than one VCPU, it raises an important question: how to select a VCPU to provide additional budget? We shall use the following example to answer this question and to illustrate ERTDS scheduler.

TABLE I
The Parameter Settings of Example 3.

VCPU	period	budget	guaranteed utilization
$vcpu_{1,1}$	8 μs	2 μs	0.25
$vcpu_{2,1}$	12 μs	6 μs	0.5
$vcpu^*$	24 μs	6 μs	0.25

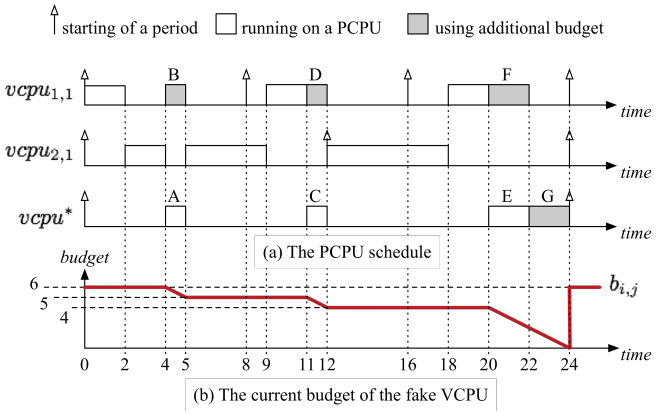

Fig. 5. An example ERTDS schedule.

Example 3: Consider a Xen virtualization system consists of two VCPUs $vcpu_{1,1}$ and $vcpu_{2,1}$ which belong to vm_1 and vm_2. The parameter settings (including periods and budgets) of these two VCPUs and the fake VCPU $vcpu^*$ are given in Table I. For ease of discussion, we assume that there is only one PCPU in the system. As a result, the G-EDF will deteriorate to the *uniprocessor earlier deadline first* (U-EDF) scheduler [26]. Note that the period and the budget of the fake VCPU $vcpu^*$ is set as 24 μs (which is the LCM of the periods of all VCPUs) and 6 μs. This means that the fake VCPU $vcpu^*$ can provide up to 6 μs additional budget to the two VCPUs for every 24 μs. Figure 5(a) shows the schedule of these VCPUs under ERTDS.

We examine some special time points as follows: initially,

at time 0, the budget of $vcpu_{1,1}$, $vcpu_{2,1}$ and $vcpu^*$ are set as 2, 6 and 6, respectively. Since $vcpu_{1,1}$ has the earliest deadline, it is selected to be running on the PCPU. At time 2, the budget of $vcpu_{1,1}$ is depleted. According to the U-EDF, $vcpu_{2,1}$ starts to run on the PCPU. Suppose that $vcpu_{2,1}$ becomes idle at time 4. Therefore, the fake VCPU $vcpu^*$ becomes the highest priority VCPU and its budget is provided to $vcpu_{1,1}$ such that $vcpu_{1,1}$ can be running on the PCPU even its budget has been depleted. As the result, the response time of $vcpu_{1,1}$'s jobs can be shortened.

We also assume that $vcpu_{2,1}$ becomes activated at time 5. Since $vcpu_{1,1}$ is running on the PCPU with the lowest priority (i.e., the priority of the fake VCPU), it will be preempted by $vcpu_{2,1}$. Note that, in Figure 5(a), the white box with the label A is to represent $vcpu^*$'s budget is used. The gray box with the label B is to represent the additional budget is used by $vcpu_{1,1}$. Also note that the Figure 5(b) shows the value of $vcpu^*$'s budget. At time 8, $vcpu_{1,1}$'s budget is replenished. However, it is not the highest priority VCPU, thus $vcpu_{2,1}$ continues its execution until its budget is depleted at time 9. Thus, $vcpu_{1,1}$ starts its execution at time 9. Later, at time 11, the budget of $vcpu_{1,1}$ is also depleted.

Since both the budget of $vcpu_{1,1}$ and $vcpu_{2,1}$ are depleted at time 11, $vcpu^*$ becomes the highest priority VCPU again. Although both $vcpu_{1,1}$ and $vcpu_{2,1}$ might use the additional budget from the fake VCPU, ERTDS selects the one which has the latest deadline, i.e., $vcpu_{1,1}$, to be running by using the additional budget. We have discussed the reason in Example 2. It greatly improves the performance since the response time of jobs can be shortened as much as possible. Once again, the white box labeled by C and the gray box labeled by D represent the additional budget from $vcpu^*$ is used and it is used by $vcpu_{1,1}$.

Later at time 18 and 20, the budget of $vcpu_{2,1}$ and $vcpu_{1,1}$ are depleted. We also assume that $vcpu_{2,1}$ and $vcpu_{1,1}$ become idle at time 18 and 22. As a result, $vcpu_{1,1}$ uses the additional budget from time 20 as shown by the white box labeled by E and the gray box labeled by F. Since $vcpu_{1,1}$ is idle from time 22, there is no any other VCPU can use the additional budget, thus the gray box labeled by G represents the $vcpu^*$'s budget did not be used by any VCPU. It can be considered that $vcpu^*$'s budget is used by itself. Note that all VCPUs' budget (including the fake VCPU) are replenished at time 24. □

Example 3 shows that the fake VCPU can provide additional budget to $vcpu_{1,1}$ so that its performance can be improved. Note that ERTDS always selects the VCPU with the latest deadline among the VCPUs in the $depletedQ$. It is because the performance of the VCPU with the latest deadline can be improved greatly than other VCPUs (as shown in Example 3).

IV. Properties

In this section, the properties of our proposed ERTDS will be discussed for uniprocessor[2] and multi-core processor environments. When uniprocessor Xen virtualization system is considered, the G-EDF scheduling algorithm is downgraded to the uniprocessor EDF scheduling [26]. The uniprocessor periodic real-time task scheduling problem [26] is to schedule a set of tasks without violating their timing constraints (i.e., deadlines). Consider the following theorem:

Theorem 1: (Liu and Layland [26]) For a uniprocessor system, a set of periodic real-time tasks $\mathcal{T} = \{\tau_i, \tau_2, \cdots, \tau_n\}$ can be scheduled by EDF if and only if

$$\sum_{\tau_i \in \mathcal{T}} \frac{C_i}{P_i} \leq 1 \tag{1}$$

where C_i and P_i are the worst-case computation time and the period of task τ_i.

Theorem 1 helps us to develop the following theorem:

Theorem 2: When RTDS is adopted for a uniprocessor Xen virtualization system, it guarantees that each $vcpu_{i,j} \in \mathbf{VCPU}$ can be running on a feasible $PCPU$ up to $B_{i,j}$ μs for every $P_{i,j}$ μs if and only if

$$\sum_{vcpu_{i,j} \in \mathbf{VCPU}} \frac{B_{i,j}}{P_{i,j}} \leq 1 \tag{2}$$

Proof: This theorem can be proven by transforming the VCPU scheduling problem into the uniprocessor periodic real-time task scheduling problem. In particular, each VCPU $vcpu_{i,j}$ can be transformed into a corresponding periodic real-time task τ_i, where the worst-case computation time and the period of τ_i are set as the budget B_i and the period P_i of $vcpu_{i,j}$. After we transformed the problem into real-time task scheduling problem, this theorem follows directly from Theorem 1. ■

When our proposed ERTDS is considered, the following lemma shows the activation of the fake VCPU:

Lemma 2: Under ERTDS, the fake VCPU $vcpu^*$ will be selected to be running on a feasible PCPU only if all VCPU have depleted their budgets, i.e., $b_{i,j} = 0$, $\forall vcpu_{i,j} \in \mathbf{VCPU}$.

Proof: Since ERTDS is based on RTDS and G-EDF is the scheduling policy for selecting a VCPU, it always select the VCPU with the earliest deadline among all VCPUs in the $runQ$. The fake VCPU $vcpu^*$ is selected only if its deadline is earlier than all other VCPUs in the $runQ$. Note that the period of the fake VCPU is the hyperperiod of all VCPUs. The only possibility that ERTDS selects the fake VCPU $vcpu^*$ is because it is the only one VCPU in the $runQ$. Thus, it is implied that all VCPUs have depleted their budgets. ■

Since ERTDS only activates the fake VCPU at the time that there is no any activated VCPU, it will not affect the guaranteed PCPU capacity of VCPUs. However, ERTDS must guarantee the PCPU capacity of all VCPUs even if it will provide additional budget to a VCPU (which has been depleted its budget). Therefore, the following theorem shows the global budget (the additional budget from the fake VCPU) must be considered:

[2]In this paper, the term 'uniprocessor' represents a processor which has only one processing core.

Theorem 3: When ERTDS is adopted for a uniprocessor Xen virtualization system, it guarantees that each $vcpu_{i,j} \in \mathbf{VCPU}$ can be running on a feasible $PCPU$ up to $B_{i,j}$ μs for every $P_{i,j}$ μs if and only if

$$(\sum_{vcpu_{i,j} \in \mathbf{VCPU}} \frac{B_{i,j}}{P_{i,j}}) + \frac{B^*}{P^*} \leq 1 \quad (3)$$

Proof: This theorem can be proven in a similar way to the proof of Theorem 2. Every VCPU $vcpu_{i,j} \in \mathbf{VCPU}$ is transformed into a corresponding periodic real-time task $\tau_i \in \mathcal{T}$, where the worst-case computation time C_i and the period P_i of a task τ_i are set as the corresponding budget B_i and the period P_i of $vcpu_{i,j}$. Note that the fake VCPU $vcpu^*$ is also transformed into a corresponding task τ^*, where its worst-case computation time and period are defined as C^* and P^*, respectively, where their values are set as the corresponding budget B^* and the period P^* of the fake VCPU $vcpu^*$. According to Theorem 1, the transformed task set $\mathcal{T}$ and τ^* are schedulable if $(\sum_{\tau_i \in \mathcal{T}} \frac{C_i}{P_i}) + \frac{C^*}{P^*} = \sum_{\tau_i \in \mathcal{T} \cup \{\tau^*\}} \frac{C_i}{P_i} \leq 1$. Thus, this theorem is proved. ■

As astute readers might pointed out that Theorem 3 is not correct since ERTDS violates the EDF policy, i.e., the additional PCPU capacity is provided to the VCPU which has latest deadline. However, it is correct when the following facts are considered: (1) the additional PCPU capacity executed by the VCPUs is also the guaranteed PCPU capacity of the fake VCPU, and (2) it is only can be used at the time that the fake VCPU has the earliest deadline among all VCPUs which follows the EDF policy.

Recall that the main purpose of RTDS is to provide a guaranteed PCPU capacity to every VCPU so that the performance of VMs can be better predicted. Also recall that the design goal of our proposed ERTDS is to provide additional PCPU capacity to VCPUs when they depletes their budgets. However, according to Theorem 3, the predefined values of budget and period of all VCPUs have to be restricted such that the sum of the guaranteed PCPU capacity and the global budget (provided by the fake VCPU) does not exceed 1. Otherwise, the predefined PCPU capacity for each VCPU cannot be guaranteed. The following theorem provides the maximum value of the global budget, i.e., the budget of the fake VCPU:

Lemma 3: For a uniprocessor Xen virtualization system, the following condition must be hold such that ERTDS can guarantee the predefined PCPU capacity of all VCPU:

$$B^* \leq (1 - \sum_{vcpu_{i,j} \in \mathbf{VCPU}} \frac{B_{i,j}}{P_{i,j}}) P^* \quad (4)$$

Proof: This lemma is correct by considering that Equation (3) of Theorem 3 must be hold for guaranteeing the predefined PCPU capacity of all VCPU. ■

When multi-core environments are considered, the following theorem provides the schedulability of periodic real-time tasks scheduling problem:

Theorem 4: (Goossens, Funk, and Baruah [27]) For a mulit-core system, a set of periodic real-time tasks $\mathcal{T} = \{\tau_i, \tau_2, \cdots, \tau_n\}$ can be scheduled by G-EDF if and only if

$$\sum_{\tau_i \in \mathcal{T}} \frac{C_i}{P_i} \leq m - \max\{\frac{B_i}{P_i}\}(m-1) \quad (5)$$

where m is the number of cores in the system.

According to Theorem 4, we can derive the following theorem and lemma for mulit-core Xen virtualization systems:

Theorem 5: When ERTDS is adopted for a multi-core Xen virtualization system, it guarantees that each $vcpu_{i,j} \in \mathbf{VCPU}$ can be running on a feasible $PCPU$ up to $B_{i,j}$ μs for every $P_{i,j}$ μs if and only if

$$(\sum_{vcpu_{i,j} \in \mathbf{VCPU}} \frac{B_{i,j}}{P_{i,j}}) + \frac{B^*}{P^*} \leq m - \max\{\frac{B_i}{P_i}\}(m-1) \quad (6)$$

Lemma 4: For a mulit-core Xen virtualization system, the following condition must be hold such that ERTDS can guarantee the predefined PCPU capacity of all VCPU:

$$B^* \leq (m - \max\{\frac{B_i}{P_i}\}(m-1) - \sum_{vcpu_{i,j} \in \mathbf{VCPU}} \frac{B_{i,j}}{P_{i,j}}) P^* \quad (7)$$

V. Performance Evaluation

To evaluate the performance of our proposed ERTDS, we have implemented ERTDS in Xen version 4.7. The experimental environment was built upon a PC with an Intel Quad Core i5-6500 3.2GHz processor and 8GB RAM. We have conducted four experiments for evaluating the performance of our proposed ERTDS and the original RTDS. In particular, one of the experiments was performed based on RUBiS benchmark [28], [29] so that the performance of our proposed ERTDS could be better understood. In the rest of this section, the experimental results of uniprocessor (single core) and multi-core environments are presented in subsections V-A and V-B. Subsection V-C is the experimental results of RUBiS benchmark.

A. Experimental Results of Uniprocessor Environments

In this subsection, two experiments are presented to verify the performance of our proposed ERTDS and the original RTDS in a uniprocessor environment. The first experiment, called Experiment A, was set up as follows: 2 VMs were configured to be running on one core and each VM only has one VCPU. When these VCPUs were scheduled by ERTDS, their budgets and periods are given in Table II. Note that the sum of the guaranteed PCPU capacity satisfies Equation (3) and (4) so that it can be guaranteed by ERTDS. For comparison, we also designed different settings of VCPU's budgets and periods for the original RTDS. Since RTDS does not provide additional budget to a VCPU which has depleted its budget, its guaranteed PCPU capacity of VCPUs can be higher than that of ERTDS.

Table III shows the parameter settings for RTDS. Note that the guaranteed PCPU capacity of VCPUs in Table III satisfies Equation (2). Also note that the lower PCPU capacity of ERTDS is the price to paid for providing additional budget

TABLE II
THE PARAMETER SETTINGS OF EXPERIMENT A (ERTDS).

VCPU	$vcpu_{1,1}$	$vcpu_{2,1}$	$vcpu^*$
budget (μs)	2000	4500	4500
period (μs)	5000	10000	30000
guaranteed PCPU capacity	0.4	0.45	0.15

TABLE III
THE PARAMETER SETTINGS OF EXPERIMENT A (RTDS).

VCPU	$vcpu_{1,1}$	$vcpu_{2,1}$
budget (μs)	2350	5300
period (μs)	5000	10000
guaranteed PCPU capacity	0.47	0.53

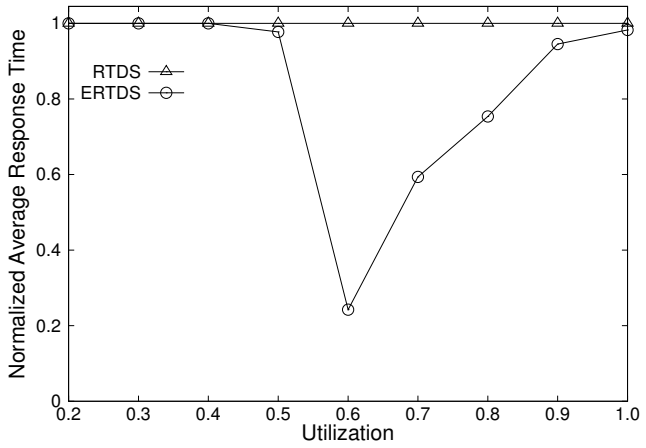

Fig. 6. Normalized average response time of Experiment A.

TABLE IV
THE PARAMETER SETTINGS OF EXPERIMENT B (ERTDS).

VCPU	$vcpu_{1,1}$	$vcpu_{2,1}$	$vcpu_{3,1}$	$vcpu_{4,1}$	$vcpu^*$
budget (μs)	1000	2000	3000	7500	4500
period (μs)	5000	10000	15000	30000	30000
guaranteed PCPU capacity	0.2	0.2	0.2	0.25	0.15

TABLE V
THE PARAMETER SETTINGS OF EXPERIMENT B (RTDS).

VCPU	$vcpu_{1,1}$	$vcpu_{2,1}$	$vcpu_{3,1}$	$vcpu_{4,1}$
budget (μs)	1200	2400	3600	8400
period (μs)	5000	10000	15000	30000
guaranteed PCPU capacity	0.24	0.24	0.24	0.28

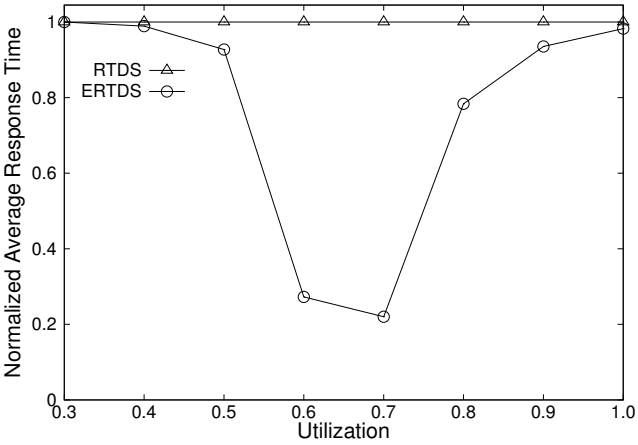

Fig. 7. Normalized average response time of Experiment B.

to VCPUs so that the performance could be improved. For the different settings of the VCPUs in Table IV and V, we randomly generated a set of tasks such that the total utilization of VCPUs varied from 20% to 100% stepped by 10%. In more detail, the utilization of $vcpu_{1,1}$ was set to 10% during the experimental time, and the utilization of $vcpu_{2,1}$ was evaluated from 0% to 80%. Such an experimental setting is to evaluate the performance of our proposed ERTDS when a VCPU's run-time requirement is higher than its guaranteed PCPU capacity. For each total utilization setting, 10 task sets were evaluated and their results were averaged.

Figure 6 shows the averaged response time of tasks. For ease of comparison, the averaged response time are normalized with respect to the results of RTDS. As shown in Figure 6, ERTDS outperforms the original RTDS. In particular, the averaged response time of ERTDS starts to decrease when the total utilization is higher than 50%, and it decreases greatly when the total utilization is higher than 60%. Astute readers might point out that the averaged response time of ERTDS increases when the total utilization is higher 70%. It is because the run-time requirements of all VCPUs were lower than their guaranteed PCPU capacities for both of RTDS and ERTDS when the total utilization is lower than 50%. Recall that the utilization of $vcpu_{1,1}$ is no more then 10%. Therefore, the utilization of $vcpu_{2,1}$ will be lower than 40% which is lower than its guaranteed PCPU capacity for both of RTDS and ERTDS (i.e., 53% for RTDS and 45% for ERTDS). However, the utilization of $vcpu_{2,1}$ will be higher than its guaranteed PCPU capacity when the total utilization is higher than 60%. Under ERTDS, a VCPU might has a chance to obtain additional PCPU capacity by using the global budget provided by the fake VCPU. In contrast, RTDS cannot obtain additional budget such that its performance will be deteriorated greatly. Therefore, our proposed ERTDS outperforms RTDS when the utilization of $vcpu_{2,1}$ is higher than the guaranteed PCPU capacity, i.e., when the total utilization is higher than 60%.

For ERTDS, the guaranteed PCPU capacity of $vcpu_{2,1}$ is 45%, and it might obtain additional PCPU capacity from the fake VCPU up to 15%. Thus, it might get up to 60% PCPU capacity for task executions. When the total utilization is higher than 70%, the utilization of $vcpu_{2,1}$ will be higher than 60%. Hence, the performance of ERTDS will be deteriorated. This is the main reason that the averaged response time of ERTDS increases when the total utilization is higher than 70%. Note that, both for RTDS and ERTDS, $vcpu_{2,1}$ cannot use the redundant PCPU capacity from $vcpu_{1,1}$. Because their predefined PCPU capacity must be guaranteed even if their requirements are lower.

The second experiment, called Experiment B, is presented to evaluate the performance of ERTDS and RTDS with more VMs in a uniprocessor environment. The settings of Experiment B is similar to that of Experiment A except the number of VMs in the system. In particular, Experiment B has 4 VMs which were configured to be running on one core and each VM has one VCPU. Table IV and Table V show the parameter settings of ERTDS and RTDS, respectively.

TABLE VI
THE PARAMETER SETTINGS OF EXPERIMENT C (ERTDS).

VCPU	$vcpu_{1,1}$	$vcpu_{2,1}$	$vcpu_{3,1}$	$vcpu_{4,1}$	$vcpu^*$
budget (μs)	1720	3440	5160	13125	8415
period (μs)	5000	10000	15000	30000	30000
guaranteed PCPU capacity	0.344	0.344	0.344	0.4375	0.2805

TABLE VII
THE PARAMETER SETTINGS OF EXPERIMENT C (RTDS).

VCPU	$vcpu_{1,1}$	$vcpu_{2,1}$	$vcpu_{3,1}$	$vcpu_{4,1}$
budget (μs)	2064	4128	6192	14448
period (μs)	5000	10000	15000	30000
guaranteed PCPU capacity	0.4128	0.4128	0.4128	0.4816

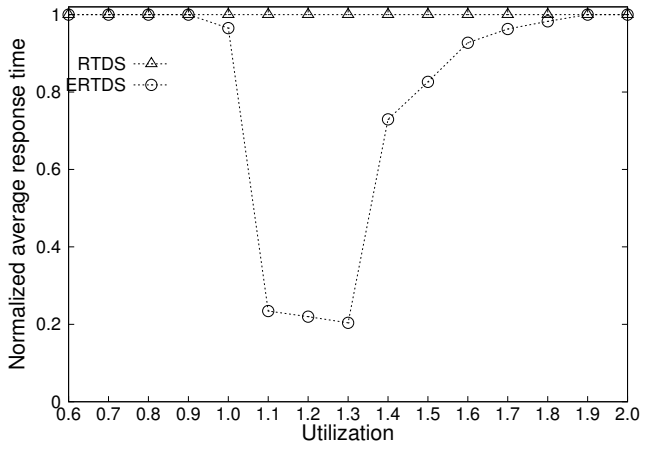

Fig. 8. Normalized average response time of Experiment C.

For the different settings of the VCPUs in Table IV and V, the total utilization of VCPUs was varied from 30% to 100% stepped by 10%. In particular, the utilization of $vcpu_{1,1}$, $vcpu_{2,1}$, and $vcpu_{3,1}$ do not exceed their guaranteed PCPU capacities, i.e., for these 3 VCPUs, each was given a fixed 10% utilization. In other words, the utilization of $vcpu_{4,1}$ was evaluated from 0% to 70%. For each total utilization setting, 10 task sets were evaluated and their results were averaged. Figure 7 shows the averaged response time of tasks. We also normalize the results with respect to the RTDS for ease of comparison.

Figure 7 shows that the performance of ERTDS outperforms RTDS in all cases, which is consistent with that of Experiment A. In particular, the averaged response time of ERTDS decreases greatly when the total utilization is higher than 50%, and it increases when the total utilization is higher 70%. However, on the one hand, the performance of ERTDS and RTDS are similar since all VCPUs have sufficient PCPU capacity for both RTDS and ERTDS when the total utilization is lower than 50%. On the other hand, the performance of ERTDS outperforms RTDS greatly when the total utilization is higher than 50%. It is because $vcpu_{4,1}$ obtained the additional PCPU capacity from the fake VCPU when its utilization is higher than the guaranteed PCPU capacity. While RTDS cannot obtain additional PCPU capacity such that its performance will be deteriorated.

B. Experimental Results of Multi-Core Environments

Our third experiment, called Experiment C, is to evaluate the performance of RTDS and ERTDS for a mulit-core environment. Similar to Experiment A and B, Experiment C was set up for 4 VMs and each has 1 VCPU but it will be running on two cores. When VCPUs were scheduled by ERTDS, their budgets and periods are given in Table VI. Note that the sum of the guaranteed PCPU capacity satisfies Equation (6) and (7) so that it can be guaranteed by ERTDS. In particular, the global budget is set according to Equation (7), were $B^* = (m - \max\{\frac{B_i}{P_i}\}(m-1) - \sum_{vcpu_{i,j} \in \mathbf{VCPU}} \frac{B_{i,j}}{P_{i,j}})P^* = ((2 - 0.4375) - (0.344 + 0.344 + 0.344 + 0.4375))30000 = 8415$. In other words, the fake VCPU can provide up to $\frac{8415}{30000} = 28.05\%$ PCPU capacity to other VCPUs. The parameter settings for RTDS is given in Table VII.

According to Theorem 4, the sum of the guaranteed PCPU capacity cannot exceed $m - \max\{\frac{B_i}{P_i}\}(m-1) = 2 - 0.28 = 1.72$. The guaranteed PCPU capacity of VCPUs in Table VII satisfies Equation (5). We randomly generated a set of tasks such that the total utilization of VCPUs varied from 60% to 200% stepped by 10%. For $vcpu_{1,1}$, $vcpu_{2,1}$, and $vcpu_{3,1}$, their utilization is fixed to 20%. Thus, the utilization of $vcpu_{4,1}$ was evaluated from 0% to 140%. We also generated 10 task sets for each utilization setting and their results were averaged. Figure 8 shows the normalized averaged response time of tasks.

Figure 8, 6 and 7 share a similar trend in the results. In particular, Figure 8 shows that ERTDS outperforms RTDS greatly when the total utilization is higher than 90%, and the averaged response time of ERTDS increases when the total utilization is higher than 140%. Note that the reason is the same as that for Experiment A. Based on the experimental results from this section, it is shown that the additional global budget provided by ERTDS improves the performance of VCPUs greatly.

C. Experimental Results Based on the RUBiS Benchmark

In this subsection, an experiment, called Experiment D, was conducted based on the RUBiS benchmark version 1.4.3 [28], [29]. RUBiS is a popular open source benchmark which is a multi-tier web-based auctioning system modeled after eBay.com. The experiment was set up as follows: 3 VMs (i.e., vm_1, vm_2 and vm_3) were configured to be running on one core and each VM only has one VCPU. The RUBiS benchmark was performed on the three VMs, where vm_1 is the Apache web server and PHP frontend, vm_2 is the backend MySQL database, and vm_3 is the RUBiS client and workload generator. When these VCPUs were scheduled by ERTDS and the original RTDS, their budgets and periods are given in Table VIII and IX, respectively.

Figure 9 shows the throughput (the number of requests completed per second) of Experiment D when the RUBiS benchmark was performed under ERTDS and original RTDS. The result shows that our proposed ERTDS outperforms the original RTDS (up to 11.68%). In particular, the throughput of ERTDS and RTDS are 86 and 77, respectively. This is because our proposed ERTDS is capable to provide additional PCPU capacity to an overcommitted VCPU (i.e., $vcpu_{1,1}$ and $vcpu_{1,2}$ for the frontend and backend servers) by using the

TABLE VIII
THE PARAMETER SETTINGS OF EXPERIMENT D (ERTDS).

VCPU	$vcpu_{1,1}$	$vcpu_{2,1}$	$vcpu_{3,1}$	$vcpu^*$
budget (μs)	3500	3000	1000	1500
period (μs)	10000	10000	5000	10000
guaranteed PCPU capacity	0.35	0.3	0.2	0.15

TABLE IX
THE PARAMETER SETTINGS OF EXPERIMENT D (RTDS).

VCPU	$vcpu_{1,1}$	$vcpu_{2,1}$	$vcpu_{3,1}$
budget (μs)	4100	3500	1200
period (μs)	10000	10000	5000
guaranteed PCPU capacity	0.41	0.35	0.24

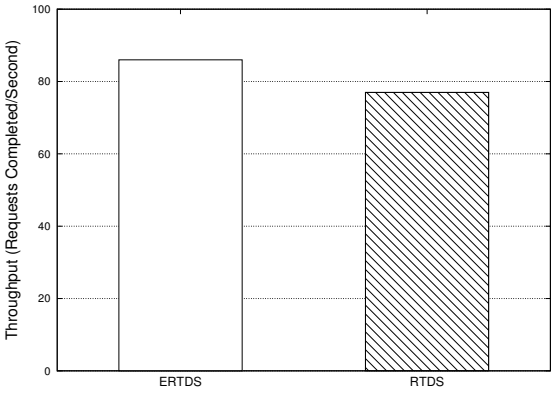

Fig. 9. Throughput of Experiment D.

global budget from the fake VCPU. However, the original RTDS cannot obtain any additional budget.

VI. CONCLUSION AND THE FUTURE WORK

In this paper, an RTDS-based CPU scheduler, called enhanced real-time deferrable server (ERTDS), is proposed for Xen virtualization systems. Under ERTDS, each VCPU has a guaranteed PCPU capacity and might has chances to obtain additional PCPU capacity at the run-time so that the performance of VMs can be improved greatly. We have implemented ERTDS in Xen 4.7 and performed a series of experiments. Our experimental results show that ERTDS outperforms the original RTDS. Our future work will focus on the analysis of performance guarantee as well as the run-time overheads.

ACKNOWLEDGMENT

The authors would like to thank Mr. Chen-Yuan Wang and Mr. Shou-Liang Sun for their help in implementation of ERTDS.

REFERENCES

[1] K. Fraser, S. Hand, T. Harris, I. Leslie, and I. Pratt, "The Xenoserver computing infrastructure," Computer Laboratory, University of Cambridge, Tech. Rep. Technical Report UCAM-CL-TR-552, January 2003.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, Bolton Landing, New York, USA, October 19 – 22 2003, pp. 164–177.

[3] Z. Chen, Y. Zhu, Y. Di, and S. Feng, "Optimized self-adaptive fault tolerance strategy in simulation system based on virtualization technology," *IAENG International Journal of Computer Science*, vol. 42, no. 4, pp. 305–312, 2015.

[4] Z. Kartit and M. E. Marraki, "Applying encryption algorithm to enhance data security in cloud storage," *Engineering Letters*, vol. 23, no. 4, pp. 277–282, 2015.

[5] M. Noorafiza, K. Ishak, H. Maeda, M. Shiratori, T. Kinoshita, and R. Uda, "Characteristic patterns of timestamps from android operating system on mobile device and virtual machine," *IAENG International Journal of Computer Science*, vol. 43, no. 2, pp. 212–218, 2016.

[6] Z. Wang, J. Wang, B. Li, Y. Liu, and J. Ma, "Online cloud provider selection for QoS-sensitive users: Learning with competition," *IAENG International Journal of Computer Science*, vol. 43, no. 3, pp. 310–317, 2016.

[7] E. Ackaouy, "The xen credit CPU scheduler," in *Xen Summit*, San Jose, CA, USA, September 7–8 2006.

[8] D. Chisnall, *The Definitive Guide to the Xen Hypervisor*. Prentice Hall, 2007, iSBN: 978-0133582499.

[9] K. Duda and D. Cheriton, "Borrowed-virtual-time (BVT) scheduling: Supporting latency-sensitive threads in a general-purpose scheduler," in *Proceedings of the 17th ACM Symposium on Operating Systems Principles (SOSP'99)*, Charleston, South Carolina, USA, December 12–15 1999, pp. 261–278.

[10] S. Xi, M. Xu, C. Lu, L. Phan, C. Gill, O. Sokolsky, and I. Lee, "Real-time multi-core virtual machine scheduling in Xen," in *Proceedings of the 4th International Conference on Embedded Software (EMSOFT)*, 12-17 October 2014.

[11] A. Makkar, "Scope and performance of credit-2 scheduler," in *Xen Summit*, Toronto, Canada, August 25–26 2016.

[12] H. Chen, H. Jin, K. Hu, and M. Yuan, "Adaptive audio-aware scheduling in xen virtual environment," in *Proceedings of the 2010 IEEE/ACS International Conference on Computer Systems and Applications (AICCSA)*, May 16-19 2010.

[13] B. Kim, J. Lee, S. Lee, and K. YW, "Low latency scheduling on multi boost environment," in *Proceedings of the International Conference on Hybrid Information Technology (ICHIT)*, 26-28 August 2010, pp. 720–724.

[14] Z. Chang, J. Li, R. Ma, Z. Huang, and H. Guan, "Adjustable credit scheduling for high performance network virtualization," in *Proceedings of the 2012 IEEE International Conference on Cluster Computing*, 2012, pp. 337–345.

[15] C. Tseng and Y. Chung, "An enhanced CPU scheduler for xen hypervisor to improve performance in virtualized environment," in *Proceedings of the International Conference of Ubiquitous Computing and Multimedia Applications (UCMA)*, Bali, Indonesia, June 2012, pp. 62–67.

[16] X. Ding, A. Xiong, and C. Yang, "Optimation of xen scheduler for multitasking," in *Proceedings of the 4th IEEE International Conference on Software Engineering and Service Science (ICESS)*, 23-25 May 2013.

[17] L. Zeng, Y. Wang, W. Shi, and D. Feng, "An improved xen credit scheduler for i/o latency-sensitive applications on multicores," in *Proceedings of the 2013 International Conference on Cloud Computing and Big Data*, 16-18 December 2013, pp. 267–274.

[18] H. Guan, R. Ma, and J. Li, "Workload-aware credit scheculer for improving network i/o performance in virtualization environment," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 130–142, April-June 2014.

[19] C. Shen, X. Liu, and W. Tong, "WARS : A workload-aware cpu resources scheduling for the cloud computing environment," *Advanced Science and Technology Letters*, vol. 63, pp. 6–11, 2014.

[20] J. Wu, C. Wang, and J. Li, "LA-Credit: A load-awareness scheduling algorithm for Xen virtualized platform," in *Proceedings of the 2nd IEEE International Conference on High Performance and Smart Computing (HPSC)*, April 2016, pp. 234–239.

[21] S. Xi, C. Li, C. Lu, C. Gill, M. Xu, L. Phan, I. Lee, and O. Sokolsky, "RT-OpenStack: CPU resource management for real-time cloud computing," in *Proceedings of the 8thIEEE International Conference on Cloud Computing (CLOUD)*, June 2015, pp. 179–186.

[22] A. Burns and R. I. Davis, "A survey of research into mixed criticality systems," *ACM Computing Surveys (SUR)*, vol. 50, no. 6, article no. 82, January 2018.

[23] J. Wu and J. Li, "ERTDS: A dynamic CPU scheduler for xen virtualization systems," in *Proceedings of the 2017 IEEE International Conference on Applied System Innovation (ICASI)*, May 13-17 2017.

[24] J. Lehoczky, L. Sha, and S. JK, "Enhanced aperiodic responsiveness in hard real-time environments," in *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, 1987, pp. 261–270.

[25] T. Baker, "A comparison of global and partitioned EDF schedulability tests for multiprocessors," Department of Computer Science, Florida State University, Tech. Rep. Technical Report TR-051101, 2005.
[26] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the Association for Computing Mahinery (JACM)*, vol. 20, no. 1, pp. 46–61, 1973.
[27] J. Goossens, S. Funk, and S. Baruah, "Priority-driven scheduling of periodic task systems on multiprocessors," *Journal of Real-Time Systems*, vol. 25, no. 2-3, pp. 187–205, 2003.
[28] E. Cecchet, J. Marguerite, and W. Zwaenepoel, "Performance and scalability of EJB applications," in *Proceedings of the 17th ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications*, 2002, pp. 246–261.
[29] OW2 Consortium, "RUBiS: Rice university bidding system," Website (available at http://rubis.ow2.org).

Jun Wu received his Ph.D. degree in Computer Science and Information Engineering from National Chung Cheng University, Chiayi, Taiwan, in 2004. Currently, he is an Associate Professor at Department of Computer Science and Information Engineering, National Pingtung University, Pingtung, Taiwan. His research interests include: (1) Energy-Efficient Task Scheduling and Synchronization for Real-Time Embedded Systems, and (2) High Performance Resource Management for Virtualization Platforms. Dr. Wu is a member of the IEEE and the IAENG. He has also been a visiting researcher in University of York, UK, and a visiting scholar in Academia Sinica, Taiwan.

Jian-Fu Li received his Master degree in Computer Science and Information Engineering from National Pingtung University, Pingtung, Taiwan, in 2017. His research interests include: (1) Xen Virtualization Systems, and (2) Real-Time Embedded Systems.