# Real-time Scheduling in a Virtualized CE Device

Seehwan Yoo, Young-Pil Kim, and Chuck Yoo,
*Dept. of Computer Science and Engineering, Korea University, South Korea*

***Abstract*—Recently, virtualization has been widely adopted in embedded systems. Because most of embedded systems have real-time requirements, real-time property needs to be preserved in a virtual machine system. We present how we can deal with real-time issues in an embedded virtual machine. Our contribution is that we provide a new *abstract periodic interface* to a real-time virtual machine so that the virtual machine can meet the physical execution condition.**

## I. INTRODUCTION

Recent embedded systems are evolving to smart, secure personal devices. For security reasons, virtual machine architecture is trying to be adopted in embedded systems [3, 4]. Because the virtualization layer implements strong isolation among execution environments, it provides a convenient way to secure user data, code, and important personal assets without modifying the previous software including the operating system stacks. Because CE devices are becoming more personalized, security issue is one of rising concerns. Virtual machine system is a good design practice to secure personal assets with the smallest changing cost.

However, real-time support in an embedded virtual machine system is challenging. Because the virtualization hides the physical hardware information under the virtualization layer, and the guest OS can access only resources provided by the VMM (Virtual Machine Monitor). In case of processor time, the VMM provides a virtual processor which is a time-shared physical processor. Moreover, a guest OS over a virtual machine cannot timely see the physical time.

One of the most fundamental and significant questions on real-time problems is that whether the virtualized system can keep its original properties. Namely, can we use the previously used real-time system parameters instead of finding new real-time parameters such as tasks period and execution time? If it is not possible, the important property of virtualization, i.e. application reuse without any modification, is lost, and an amount of redundant engineering effort is required.

Therefore, this paper proposes a possible solution to deal with the problem in real-time scheduling in a virtual machine system. With the help of compositional scheduling framework, we can analyze virtual machine level real-time schedulability. Using the compositional scheduling, real-time applications in CE devices can be easily take the advantage of virtualization.

## II. REAL-TIME SCHEDULING IN A VIRTUAL MACHINE SYSTEM

Real-time scheduling in a virtual machine has been addressed in several research works. In [5], the author claims that workload classification reduces the complexity of scheduling in heterogeneous virtual machines. It can be applicable to well-separated virtual machine, but, in real-world workload, VM classification is not easy, and heavily depends on manual settings.

Hierarchical scheduling framework [6] deals with the fundamental problems in real-time scheduling and system composition rule. Based on supply/demand bound function, it calculates the theoretical bound of the interface utilization.

Some commercial companies [1, 2] claim that their hypervisor supports real-time characteristics. They present performance with Linux, and technical details are not revealed.

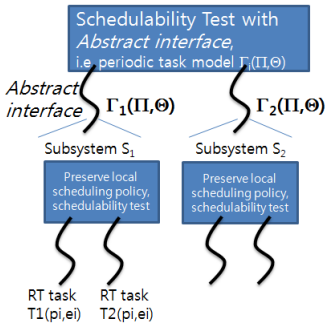


Fig. 1 Compositional Scheduling Framework

## III. COMPOSITIONAL SCHEDULING FRAMEWORK AND VIRTUAL MACHINE SYSTEM

Compositional scheduling framework provides a scheduling unit of real-time tasks. The original study focuses on embedded systems that use component-based development methodologies. Schedulers are hierarchically structured as shown in Fig. 1, and higher level scheduler schedules only components in the lower layer. The authors reveal the lower bound of the utilization that permits all the RT tasks in the sub system component schedulable.

A VMM (Virtual Machine Monitor) schedules virtual machines, and a guest OS over a VM schedules its processes or tasks. Namely, in a virtual machine system, there is a hierarchy in scheduling. Using the architectural similarity between component-based software and VM, we can take advantage of compositional scheduling in a virtual machine system. With the help of compositional scheduling framework, all the real-time tasks in a subsystem can meet the execution condition without knowledge of physical time.

It is important to choose an efficient abstract interface period because there is a hidden trade-off. In practice, efficient periodic abstract interface is hard to find.

If we choose period too large, then the worst-case execution time increases; thus the overhead increases; thus, in general, smaller period is preferred. On the other hand, if we choose too small period, the actual interface utilization can be larger than expected because the integer fraction occurs. For example, when we choose period as three time ticks, then the interface

978-1-4244-4316-1/10/$25.00 ©2010 IEEE

utilization can be given from one of 0(=0/3), 0.33(=1/3), 0.67(=2/3) or 1(=3/3). Because of the accuracy of integer-period fraction, the overhead is unavoidable.

This integer fraction overhead can be considerable in some cases. For example, in the above example, the integer fraction overhead can be up to 0.33. That means 33% of CPU utilization has to be wasted. However, we can reduce this integer-fraction overhead by carefully choosing the interface period. For example, when the interface utilization is given 0.4, then integer fraction overhead can be largely reduced from 27% to 0% by adjusting interface period from 3 to 5. Thus, if it is possible, we need to find an optimal interface period for the given component which has a real-time task set.

Now, we present an algorithm of getting the *abstract periodic interface* for real-time virtual machine scheduling. We can find the optimal periodic interface by the algorithm 1.

**Algorithm 1. Periodic interface acquisition algorithm**

**Input:** $U_w$: workload utilization,
Π, Θ: abstract interface period, execution time
AB(k): Abstraction Bound function
**Output:** minΠ, minΘ: optimal period and execution time

```
FOR all Π, Θ
  Φ:= (Θ/Π -U_w)
  Calculate k
  IF ( Φ > AB(k) )    THEN
    IF ( min Φ  > Φ )   THEN
      min Φ := Φ
      minΠ := Π
      minΘ := Θ
    ENDIF
  ENDIF
ENDFOR
```

The above algorithm does not define the maximum period and execution time; thus, its time complexity increases as period increases. Because the possible abstract interface period and execution time combination can be large, we adopt a heuristic to choose the maximum abstract interface period.

Because the average overhead from integer fraction can be bounded by1/2Π, we can increase the abstract interface period by 1/2Ov to limit the overhead be a range [0, Ov]. For a given interface period Π, average integer-fraction overhead becomes 1/2Π. Proof is omitted because of the paper length.

Another heuristic algorithm can be driven by table lookup approach. When a period value is in good range, integer-fraction overhead can be controlled efficiently. For example, if we want to limit the integer-fraction overhead by 7%, we can make a table which contains possible period, execution time settings previously.

TABLE I Integer-Fraction Lookup Table

| Utilization | Π | Θ |
|---|---|---|
| 0 | any | 0 |
| 0.2 | 5 | 1 |
| 0.25 | 4 | 1 |
| 0.33 | 3 | 1 |
| 0.4 | 5 | 2 |
| 0.5 | 2 | 1 |
| 0.6 | 5 | 3 |
| 0.67 | 3 | 2 |
| 0.75 | 4 | 3 |
| 0.8 | 5 | 4 |
| 1 | any | Π |

Example) For a scheduling unit $S_0=\{W_0,R_0,A_0\}$, where $W_0=\{T_i(P_i,E_i)\}$, $R_0=\Gamma(\Pi,\Theta)$, $A_0$=EDF, $T_1$=(50, 7), $T_2$=(75, 9), find the solution space using an abstract interface and *abstract periodic interface*.

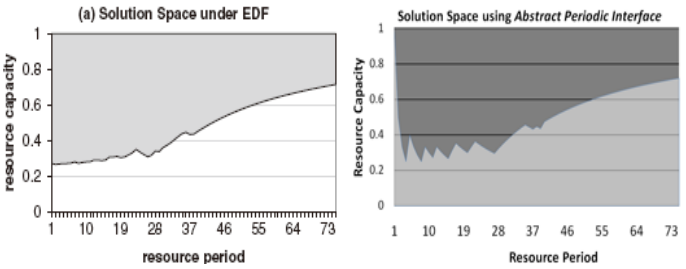


Fig. 2 Solution space under EDF using abstract interface and abstract periodic interface

Fig. 2 presents the solution space (i.e. schedulable region) under EDF scheduling for abstract interface and abstract periodic interface. We can observe that the integer-fraction overhead is not ignorable in small resource period. In addition, we can observe that even in small period, there are several points that largely reduce the integer-fraction overhead. As we can observe in the graph, the integer-fraction overhead can be considerable when the interface period is small.

## IV. Conclusion

In this paper, we introduce a real-time supporting in a virtual machine system with the compositional scheduling framework. Because of the integer-fraction overhead in time-tick based VM scheduling, compositional scheduling analysis cannot be directly applied to a virtual machine system. Therefore, we propose an interface acquisition algorithm that generates an optimal interface period and heuristics.

## Reference


[1] ArmandFrançois, & GienMichel. (2009), "A Practical Look at Micro-Kernels and Virtual Machine Monitors," Proceedings of the Consumer Communications and Networking Conference, 2009, pp. 1-7. Las Vegas, NV, IEEE.

[2] ArmandFrançois, GienMichel, MaignéGilles, & MardinianGregory. (2008), "Shared Device Driver Model For Virtualized Mobile Handsets," The Workshop on Virtualization in Mobile Computing held in conjunction with MobiSys2008, to be printed. Breckenridge, CO, ACM.

[3] BrakensiekJörg, DrögeAxel, BotteckMartin, HärtigHermann, & LackorzynskiAdam. (2008), "Virtualization as an enabler for security in mobile devices," Proceedings of the 1st workshop on Isolation and integration in embedded systems in conjunction with European Conference on Computer Systems, pp. 17-22. Glasgow, Scotland, ACM.

[4] HeiserGernot. (2009), "Hypervisors for Consumer Electronics," Proceedings of the Consumer Communications and Networking Conference, 2009, pp. 1-5. Las Vegas, NV, IEEE.

[5] RobertKaiser. (2008), "Alternatives for scheduling virtual machines in real-time embedded systems," Proceedings of the 1st workshop on Isolation and integration in embedded systems in conjunction with European Conference on Computer Systems, pp. 5-10. Glasgow, Scotland, ACM.

[6] ShinInsik, & LeeInsup. (2008), "Compositional real-time scheduling framework with periodic model," ACM Transactions on Embedded Computing Systems", 7 (3), 30:1-30:39.