Real-time Operating System Virtualization for Xen-Arm

Miri Park See-Hwan Yoo Chuck Yoo

Department of Computer and Radio Communications engineering
Korea University, Seoul, Republic of Korea
E-mail: {mrpark,shyoo,hxy}@korea.ac.kr

key words : Real-time, Embedded system, Virtualization, Xen

1. Introduction

As the virtualization technology has been growing rapidly in server consolidation fields, they become interested in its implementation on embedded systems. Xen community has released Xen Hypervisor for Arm processor which is one of the most renowned architecture in embedded systems. [1]

Xen hypervisor is designed to cooperate with para-virtualized operating systems. It means that they need to modify a guest OS to run over a virtual machine. However, there is no well-known operating system released officially for Xen-Arm so far. It is expected to port a variety of OSes including real-time OS for utilizing the virtualization on embedded systems. In this paper we present an effort for real-time operating systems to run over Xen-Arm.

2. Real-time OS para-virtualization

Para-virtualization mainly involves with modifying guest OS' source code to change hardware dependent routines to hypercalls. The hypercalls defined by Xen hypervisor allows guest OS to manage the resources. This is same when we port the real-time OS for Xen-Arm, thus we do not cover them in detail here. However, two topics should be considered carefully for some different characteristics of the real-time OS. The first issue is its memory system and the other is a timer event supported by the hypervisor.

2-1. Memory

Xen-Arm creates a page table for each guest when constructing a guest domain to support virtual memory systems. However, some real-time OSes do not use a virtual memory system but physical memory itself. They are designed to run over the limited resources for their characteristics as a SW on embedded systems.

We mapped the physical memory allocated for a real-time guest OS to the same virtual memory statically. Real-time guest OS runs as it looks physical memory but isolated by a page table provided by the hypervisor. This approach is simple and needs no modifications of guest OS to use the virtual memory systems. RTOS guest views the memories as physical ones in Figure 1.

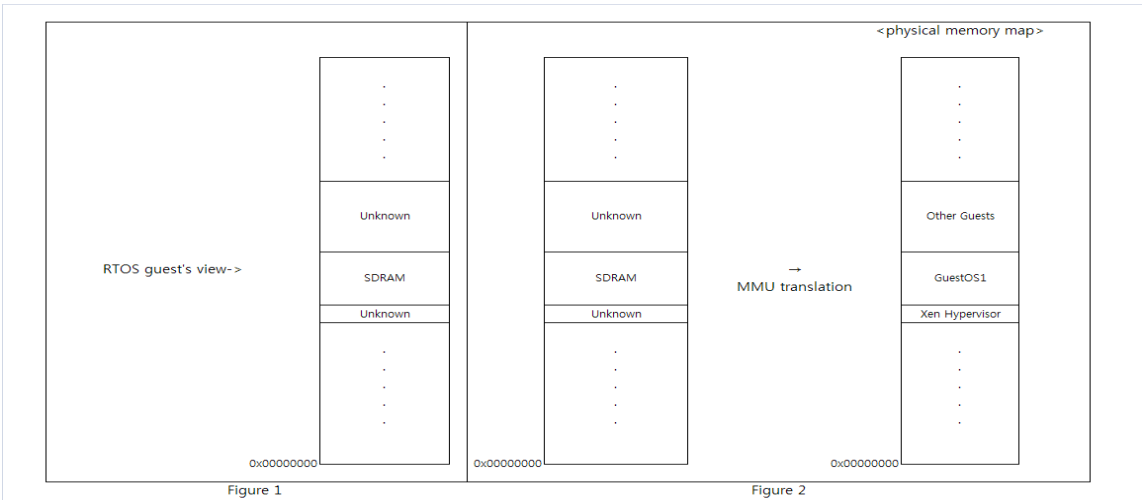


Figure 1 Figure 2

However, it is translated by MMU for domain isolation as Figure 2.

More advanced way to address this problem is to map the whole size of SDRAM as virtual addresses. In this case, hypervisor should manage the page table and handle page faults instead of the RTOS guest. It may results in a performance penalty for page faults thus we leave it to be analyzed carefully as future works.

2-2. Timer Event

Real-time kernel makes use of scheduling algorithm for real-time tasks to meet their deadlines. Some of them, such as the EDF [2](Earliest Deadline First) algorithm, need to know the physical time to decide which task to run. Thus the real-time kernel should have precise current time information.

Implementation for time managements of real-time kernel is usually accomplished by a timer interrupt. When the timer interrupt delivered, kernel updates its time and uses it for scheduling tasks. RTOS running over Xen hypervisor receives an event for a timer interrupt only when they are running. It is not a physical interrupt but virtual one sent by the hypervisor. Thus it cannot trace the real time while the guest is not running as depicted in Figure 3.

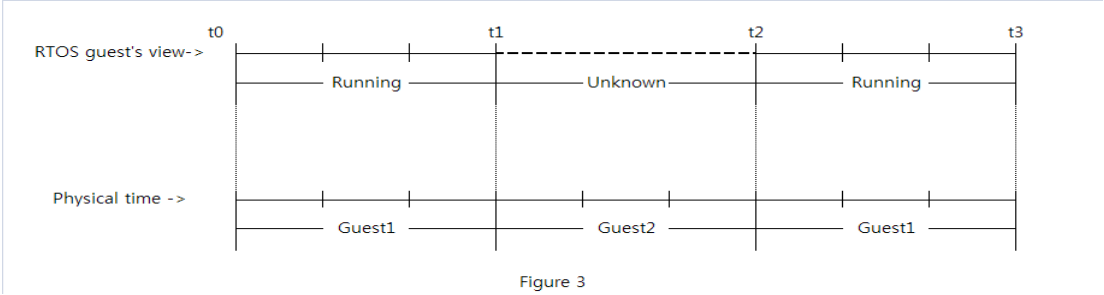
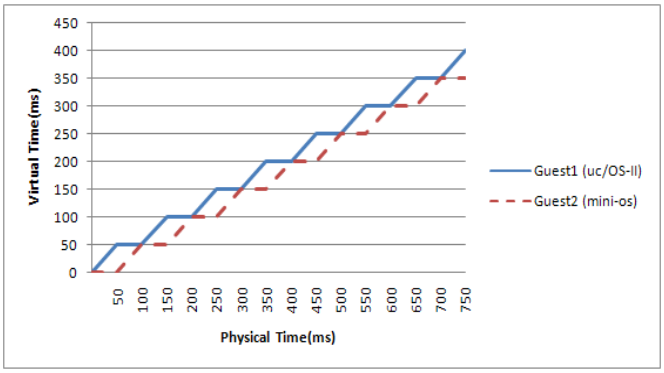


Figure 3

At t2, RTOS guest consider it receives a timer for t1. Thus its scheduler does not make a correct decision to guarantee the real-time supports. We needed to change the tick timer handler of RT guest. It should update the time passed while other guests' running. Xen provides physical time information via shared info area for guest OS to use it. However, Xen-Arm does not update it precisely in current version. Thus we modified a function *void update_dom_time(struct vcpu *v)* correctly to provide them.

3. Experiments

We have ported a free real-time OS uC/OS-II over Xen-Arm on Freescale i.MX21board (ARM926EJ-S). Graph1 shows CPU usages of two guests scheduled by bvt scheduler with same weight for each domain.



Graph 1

4. Conclusions

This paper shows a couple of issues considered when porting a real-time OS over virtual machines. Hypervisor should provide the identity-mapped page table for a guest which does not use the virtual memory system. We mapped the guest OS' physical memory space statically to the same address of virtual memory. It allows guest OS not to modify its memory programming model. The other topic is a timer management. Real-time kernel misses the timer interrupts when its virtual machine is not running. Thus we should update the precise physical time for the guaranteed service of real-time tasks.

5. References

[1] J.-Y. Hwang, S.-B. Suh, S.-K. Heo, C.-J. Park, J.-M. Ryu, S.-Y. Park, and C.-R. Kim, "Xen on arm: System virtualization using xen hypervisor for arm-based secure mobile phones," Consumer Communications and Networking Conference, 2008.

[2] 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.