Research on Real-time Virtualization Technology for Control System

Xiangying Kong
Department of Electronic Equipment, Jiangsu Institute of Automation, Lianyungang, China
iamkxy@aliyun.com

Xinran Kong
School of Mathematics and Statistics, Xi'an Jiaotong University, Xi 'an, China
kongxr_xj@aliyun.com

ABSTRACT

In recent years, the combination of multi-core processor and virtualization technology promotes the rapid application of virtualization technology in the field of real-time control. While virtualization technology gives full play to the performance of multi-core processors and improves the robustness and flexibility of application systems, the real-time performance loss caused by application virtualization has always been concerned by people. On the basis of discussing the concept of real-time control system, the principle of virtualization technology and the existing approaches of real-time virtualization technology, the real-time connotation of the virtualized control system is defined. This paper analyzes the real-time and applicable scenarios of two application schemes, namely, large-scale virtualization based on private cloud and single-node virtualization based on HyperVisor, the related real-time virtualization technology and application scenarios are discussed, and builds a test environment, and measures the interrupt response time and event response time of the corresponding schemes. The former can only meet the application with millisecond real-time performance, while the latter still has sub-microsecond real-time performance, which can meet most real-time system requirements.

CCS CONCEPTS

• **Computer systems organization**; • **Real-time systems**; • **Real-time operating systems**;

KEYWORDS

Control system, Virtualization, Real-time, Interrupt response time, Network delay

ACM Reference Format:
Xiangying Kong and Xinran Kong. 2021. Research on Real-time Virtualization Technology for Control System. In *The 5th International Conference on Computer Science and Application Engineering (CSAE 2021), October 19–21, 2021, Sanya, China.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3487075.3487080

1 INTRODUCTION

As a new computing mode, virtualization technology has good openness and flexibility, which is beneficial to improve equipment utilization and reduce enterprise costs[1,2]. In recent years, virtualization technology has rapidly expanded to many computing fields and is accelerating. Some real-time fields, such as mobile communication [3–5], online games [6, 7], industrial control [3, 8, 9], and even weapons and equipment [10, 11] have begun to try to adopt virtualization technology.

Real-time is an important indicator concerned by the control field, In view of the real-time problem of virtualization technology, some enterprises and scholars have carried out research on the improvement and promotion of virtualization real-time from different angles [12–16], and given some solutions, However, most of the existing researches are oriented to soft real-time applications, and some researches are not accurate in paying attention to real-time. Kai Chen et al. analyzed and evaluated how KVM affects the interrupt response time of VxWorks as a guest operating system[17], and Hyoseung Kim et al. optimized the real-time process of virtual interrupt processing[18], but the real-time nature of these discussions is defined as the response time of virtual machines to internal interrupts, which is not applicable to industrial real-time control systems. Starting from the real-time nature of control field, this paper analyzes and tests the real-time nature of two existing virtualization application schemes, and gives relevant suggestions.

2 PRINCIPLE OF VIRTUALIZATION TECHNOLOGY

The virtualization technology is divided into two types: Type1 Para Virtualization and Type2 Full Virtualization [19]. See figure 1a and figure 1b respectively.

Para-virtualization runs directly on the physical hardware platform, It virtualizes key hardware devices, provides several independent partitions, and provides basic services of inter-partition control and communication. Because the virtual environment of the Hypervisor is close to the hardware and can directly use the hardware resources, the performance of the para virtualized VM is basically equal to the hardware performance, which is more suitable for real-time systems. Type1 implementation mainly includes Xen, VMWare ESX, WindRiver HyperVisor, XtratuM and so on.

Full virtualization runs on the host operating system (Host OS) of the physical hardware platform. Because it runs in the traditional operating system environment, the fully virtualized HyperVisor is usually a specific software on top of the host operating system. Run related hardware, such as network adapters and other peripheral devices, through full simulation, Full virtualization allows running

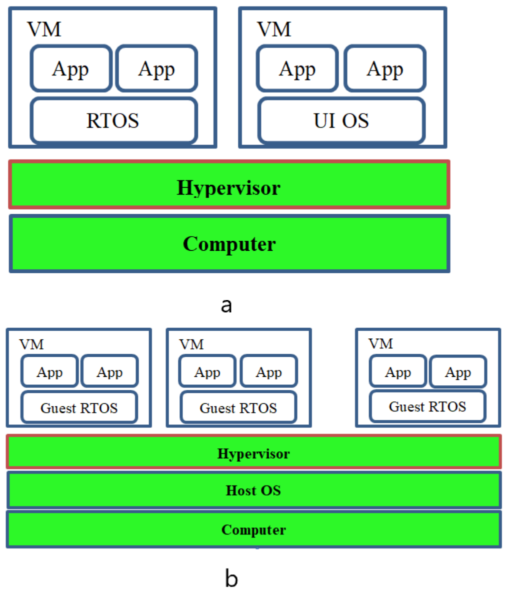

Figure 1: Type 1 and Type 2 Hypervisors.

unmodified Guest OS. In this way, it is easy to run multiple operating systems or even heterogeneous operating systems on the same hardware. Full virtualization needs to intercept special privileged instructions (such as accessing peripheral registers) executed by the kernel of guest operating system, which leads to the hypervisor falling into traps constantly, which brings great performance cost. Type2 mainly includes KVM, VMware workstation, VirtualBox, Oracle virtual machine server, etc.
Para virtualized Hypervisor such as Xen run on bare metal, which reduces the overhead of full system virtualization and is more suitable for real-time cloud computing, However, due to the fact that full virtualization does not require any changes to the guest operating system, the full virtualization technology represented by KVM has developed faster in recent years.
The full name of KVM is Kernel-based Virtual Machine, which is a virtual machine management module added to Linux kernel, It reuses the functions of process scheduling, memory management, IO management and so on in the kernel, and serves as a Hypervisor that can support virtual machine running [19]. Actually, a KVM virtual machine VM is just a process in Host Linux, which is dispatched by Host OS together with other processes.

3 TWO VIRTUALIZATION REAL-TIME APPLICATION SCHEMES

At present, there are mainly two kinds of virtualization applications. Scheme 1, as shown in figure 2, is for single-node real-time applications, making full use of the powerful processing capability of multi-core processing, and integrating real-time control applications and non-real-time applications into the same computer by adopting para virtualization technology (a few of which also adopt full virtualization technology). Compared with the traditional control system, this mode is to replace the traditional low-performance processor with a multi-core processor with powerful computing power, and

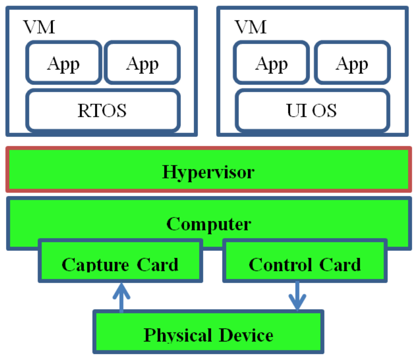

Figure 2: Schematic Diagram of System Solution for Single-Node Real-Time Application.

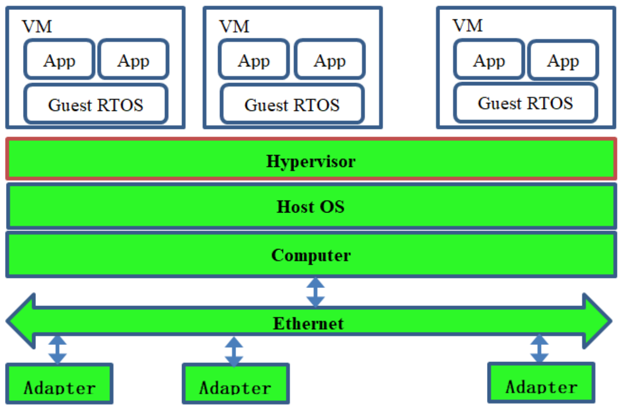

Figure 3: Schematic Diagram of System Solutions for Large-Scale Computing and Control Applications.

replace the real-time operating system with a hypervisor-based real-time operating system and a non-real-time operating system. Scheme 2 is for large-scale computing and control systems, which uses the idea of cloud computing for reference, separates computing and control, concentrates computing in a virtualized environment, uses adapters to connect with the physical world, and uses network connections between adapters and computing centers, usually using full virtualization technology. See figure 3

4 REAL-TIME ANALYSIS

4.1 Real-Time Performance of Control System

Real-time is deterministic, that is, the system can respond to external events and execute certain actions within a certain time.
Certainty is a measure of the change of system response time to specific events, and jitter is an index of real-time performance, As shown in Figure 4, it is the worst-case execution time minus the best-case execution time. Different applications have different periods and require different response times, with periods ranging from tens of microseconds to hundreds of milliseconds and acceptable jitter ranging from several microseconds to several milliseconds.

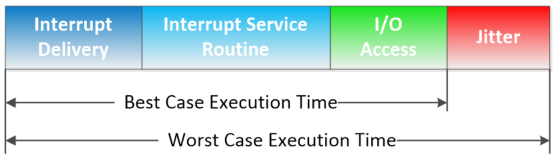

Figure 4: Event Response Time Composition.

Generally speaking, for software, the real-time performance of the system depends on Interrupt Response Time (IRT) and Context Switch Time, CST), and the calculation methods are as follows (1) and (2) respectively. These two indexes are also important goals of real-time operating system design. When interrupt nesting and task preemption are not considered, these two times are certain, which are mainly related to the instruction set that needs to be executed when corresponding operations occur. Generally, the interrupt response time of the real-time operating system is between several and ten microseconds, and the jitter is about 1-2 microseconds, which is related to the implementation mode and CPU frequency.

TimeIRT=t2-t1 (1)

TimeCST=t3-t2 (2)

The response to external events in a virtualized environment is more complex. Note that we are concerned about external events, while the existing researches on the real-time performance of virtual machines mostly focus on the real-time performance of internal interrupts (such as clock interrupts) of virtual machines, We believe that industrial control systems are Cyber-Physical Systems (CPS), and we should pay more attention to the response to the physical world.

4.2 Real-Time Analysis of Scheme 1

In Scheme 1, the system is equipped with special boards with acquisition function and control function, and the computer system senses external events or states through the acquisition board, makes calculations and judgments, and then responds through the control board. Compared with non-virtualized environment, in virtualized environment, interrupt delay is composed of hardware circuit for receiving interrupt signal, operating system for scheduling interrupt and interrupt service routine for processing interrupt request, When Guest OS for processing interrupt needs to be awakened, the context switching time that hypervisor must execute is also a considerable delay source, How to avoid or minimize this time is also an important goal of real-time optimization design of virtualized system.

Especially when adopting full virtualization technology, the virtual machine is a process of the Host OS, When the interruption occurs, the corresponding virtual machine may need to be scheduled, and the Host OS cannot see the internal process information of the virtual machine, so the response process may be interrupted by other processes in the Host OS (including other virtual machines), Therefore, even if the priority of even hander process is the highest in the virtual machine, the whole response process is uncertain.

To prevent virtual machines that need to handle interrupts from being preempted by other processes, core affinity technology provided by hardware and device passed through technology can be adopted.

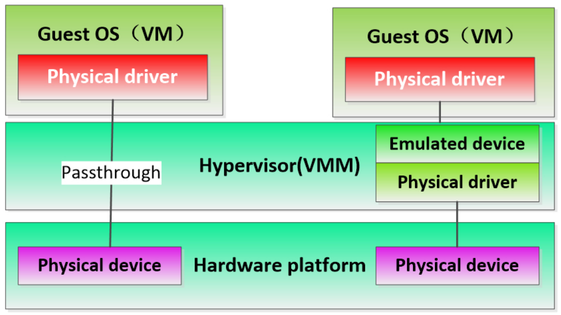

Figure 5: Equipment Passed through Mode.

As shown in Figure 5, passed through can not only make the virtual machine monopolize the hardware, but also improve the efficiency of the virtual machine using the hardware. This is because input/output simulation is the biggest performance challenge in virtual machines, Up to now, in all virtualization technologies, virtual machine monitor (VMM) must participate in all interactions between guest software and input/output peripherals. In hardware-assisted virtualization, this leads to a large number of expensive virtual machines exiting. This is because in the passed through mode, when the Guest OS interacts with peripheral devices, VMM software is not required to participate, thus avoiding a large number of virtual machine exit/entry operations.

The Linux kernel supports task affinity and interrupt affinity. For task affinity, you can call the system call sched_setaffinity () or use tools such as taskset/cpuset from the shell to set the CPU binding of the process. For interrupt affinity, Linux provides a user interface through /proc/IRQ/< IRQ _ number >/SMP_affinity file, which is used to set the CPU affinity of interrupts [20].

4.3 Real-Time Analysis of Scheme 2

Scheme 2 is actually a control system adopting cloud computing mode, The original control system must separate calculation from control, and an Adapter Process (AP) needs to be introduced to realize the connection between the physical world and cloud computing devices, as shown in Figure 3 . Cloud computing focuses on computing, communicating with adapters through the network, and realizing the collection and control of physical world information. Therefore, the real-time performance of the system is transformed into the certainty of the network message exchanged with the adapter, that is, the calculation result must be sent to the adapter at a certain time, so that the AP can complete the response to the physical world within the required time. Due to the limited number of physical network cards, the passed through mode of network cards is not realistic in practical application, Virtual machines usually communicate with the outside by means of a tap/tun device, and the response flow based on TAP/TUN device is shown in Figure 6 [21]. It can be seen that this will involve switching between kernel mode and user mode many times, and the intermediate process is easily interrupted by other processes.

The real-time performance of this kind of system depends on the following factors: 1) the time from the physical network card receiving the message to entering the interrupt handler of the Host

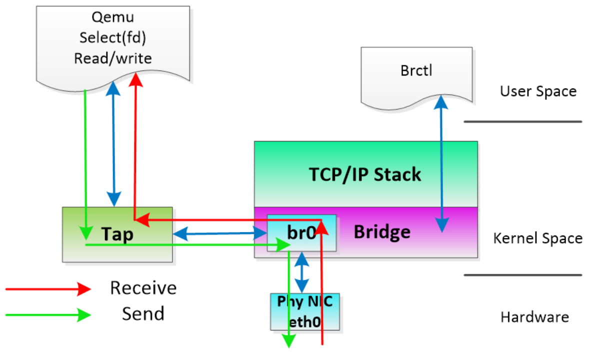

Figure 6: Schematic Diagram of Data Exchange between Virtual Machine and External Network Based on Tap/Tun.

OS network card driver depends on the interrupt response time of the Host OS; 2) The time when the 2)Host OS copies the message to the Tap device; 3) The time for 3)Qemu to read messages from Tap equipment involves the switching between kernel mode and user mode and the exit and entry time of a large number of KVM; 4) The time when the message inside the virtual machine is submitted to EHP involves the switch from kernel mode to user mode; 5) After processing, return the corresponding message according to the original route.

Scheduling execution of time handlers depends not only on the real-time performance of the hosted Guest OS, but also on the real-time performance of the hosted Host OS. See figure 2In virtual environment, the scheduling timing of application A depends on two-level scheduling, First, the Host OS schedules the Guest OS hosted by application A, and then the Guest OS schedules application A, so that application A can get the opportunity to execute. Because the Host OS doesn't know the priority of the applications in the Guest OS, once the virtual machine where the Guest OS is located is suspended due to resource competition, all its internal programs will not get the opportunity to execute. It should be pointed out that the Host OS takes on a lot of management work, and also needs to take up a lot of CPU time and manage multiple interrupt resources, which brings great uncertainty to the execution of virtual machines.

4.4 Real-Time Improvement Measures of Virtualization System

In order to improve the real-time performance of the virtualized system, the following measures can be taken:

1) both 1)Host OS and Guest OS adopt real-time operating systems to reduce the uncertainty of the operating systems themselves; Host OS usually uses Linux, In the standard Linux kernel, RT-Patch can solve the real-time problem of Linux kernel [22, 23].

2) By adopting the affinity technology provided by hardware, the virtual machine performing critical tasks is bound with the core and monopolizes the core, so as to avoid being preempted and interrupted by other processes (including other virtual machines).

3) The network card is bound to the virtual machine by passed through to improve the network communication performance of the corresponding virtual machine.

Table 1: Test Result of Interrupt Response Time of Real-Time Operating System under Physical Computer (Unit Microsecond)

OS	Minmum	Maximum	Average
VxWorks 6.8	4.17	4.42	4.24
JARI-Works 3.2	4.38	5.27	4.53

4) Set the task priority of all virtual machines to the highest to reduce the preemption of other processes (non-virtual machines).

These technologies can improve the certainty of interrupt, task scheduling and communication in the designated virtual machine, However, due to the limitation of the number of physical host processor cores and network cards, these measures can only ensure the real-time performance of a limited number of applications, and too many bindings also reduce the flexibility of the system, thus losing the original intention of adopting virtualization technology.

5 PERFORMANCE TEST

In order to test the interrupt response time, we made a clock board with a built-in counter, The interrupt period of the clock board can be set from 1-1000 milliseconds, and an interrupt is generated at the beginning of each period, At the same time, the internal counter is reset to count from 0, the counting unit is 10 nanoseconds, and the count can be read by a 32-bit register.

5.1 Real-Time Test of Physical Computer

The test environment uses Intel Core ™ i7 processor, the operating system uses VxWorks6.8 and JARI-Works3.2 respectively, the interrupt is generated by the clock board, the interrupt period is set to 10ms, the interrupt handler reads the counter value of the clock board, and tests for 10,000 times, The test results are shown in Table 1

5.2 Performance Test of Scheme 1

We built test environments for semi-virtualization and full virtualization. Hardware adopts Intel Core i7 processor+clock board.

Wind River Hypervisor is adopted for semi-virtualization, and VxWorks 6.8 is adopted for Guest OS.

The fully virtualized Host OS adopts Linux+KVM+RT Patch, and the Guest OS adopts VxWorks 6.8 and JARI-Works 3.2 . The test virtual machine adopts kernel affinity setting, and the clock board adopts passed through mode and is directly managed by the operating system of the tested virtual machine. Interrupt response time of the tested Guest OS interrupt handler, the tested Guest OS clock board interrupt handler reads the clock board counter value and tests it for 10,000 times, The test results are shown in Table 2It can be seen that compared with the traditional physical machine, the interrupt response time of the system is increased by 4-16 microseconds, and the jitter time is increased by 2-3 microseconds, which can still meet the requirements of most strong real-time applications.

Table 2: Test Result of Interrupt Response Time of Real-Time Operating System of Virtual Platform (in Microseconds)

Host OS	Guest OS	Minmum	Maximum	Average
Wind River Hypervisor	VxWorks 6.8	8.36	10.22	8.85
Linux+KVM+RT Patch	VxWorks 6.8	18.67	20.74	19.31
	JARI-Works 3.2	18.32	21.85	19.76

Table 3: Test Result of Virtual Platform Adapter Event Response Time (in Microseconds)

Host OS	Guest OS	average	maximum
CentoOS+KVM	Linux	156	5632
+RT Patch	JARI-Works 3.2	151	4671

5.3 Performance Test of Scheme 2

Server adopts eight Inspur Yingxin NF 5280 M 4 (CPU Intel Xeon E5-2630), host OS is CentOS+KVM+RT Patch, Guest OS is VxWorks 6.8, JARI-Works 3.2, Linux; The adapter adopts a PC (Intel Core i7 processor+clock board, the operating system is VxWorks 6.8, and the clock board. 10 gigabit Ethernet card Intel I350, gigabit Ethernet card Intel 82599ES.
Test method: the test virtual machine adopts the nuclear affinity setting; The interrupt period of the adapter clock board is set to 10ms,Every time an interrupt is generated, the adapter sends a frame message with increasing sequence number to the Guest OS in the server, totaling 100,000 frames, and records the time of sending the sequence number message in the array T1[] (subscript is equal to the frame number); After receiving the message, the Guest OS in the server immediately replies the corresponding message (including serial number) to the adapter; After receiving the reply message, the adapter records the receiving time in the array T2[] (the subscript is equal to the frame number). The test results are shown in Table 3 It can be seen that compared with the traditional physical machine, the time response time of the system is increased by 140-150 microseconds, and the jitter time is increased to 4-5 milliseconds, which can only meet the requirements of applications with millisecond jitter time, and is not suitable for most strong real-time systems.

6 CONCLUSION

In view of the real-time problem of virtual technology, we analyzed the virtual technology principle and application response requirements, discussed the relevant real-time optimization measures, and tested the common solution building environment. Test results show that scheme 1 can meet most control system requirements, while scheme 2 can only meet the application requirements with millisecond jitter time. It should be pointed out that the above tests have not considered the system load.

ACKNOWLEDGMENTS

This research work was funded by National Key R&D Program of China 2018YFB1305900.

REFERENCES

[1] W. Huang, J. Liu, B. Abali, D. Panda (2006). A case for high performance computing with virtual machines, in: Proc.of 20th Annual ACM Int'l Conference on Supercomputing, pp. 125-134.
[2] M. Mergen, V. Uhlig, O. Krieger, J. Xenidis (2006). Virtualization for high-performance computing, in: Proc.of ACM SIGOPS Operating Systems Review, vol. 40 (2), pp. 8-11.
[3] Xia Hao, Liu Qin (2014). Avenue to Jane – Cloud NF, ZTE's mobile network virtualization solution, No.1, 18-19.
[4] Multi-Core with virtualization, a solution for future smart phones. http://www.alphagalileo.org , Aug 2010 .
[5] S. Yoo, Y. Liu, C. Hong, C. Yoo and Y. G. Zhang (2008). MobiVMM: a Virtual Machine Monitor for Mobile Phones, Proc. of the 1st Workshop on Virtualization in Mobile Computing, Breckenridge, USA, pp.1-5.
[6] https://www.intel.cn/content/www/cn/zh/virtualization/virtualization-technology/intel-virtualization-technology.html 2021.4.30.
[7] https://cloud.tencent.com/developer/information/all.html 2021.4.30.
[8] The virtualization technology of industrial controller-edge computing entered the factory workshop, http://www.eepw.com.cn/article/202008/416580.htm, April 30, 2021.
[9] WindRiver, virtualization requirements of next generation industrial control system, http://windriver.com.cn/downloads/files/WP-requirements-virtualization-next-gen-industrial-cs-white-paper-cn.pdf 2021.4.30 .
[10] Zhang Junhong, Tong Qiang (2019). A New Generation of Airborne Software Design Based on Software Virtualization Technology, Journal of Nanjing Aerospace University, Vol.51, No.6, 772-777.
[11] Xiong Huagang (2014). Jake, Architecture and Key Technologies of Avionics Cloud, International Aviation, No.7, 59-60, 2 pages in total.
[12] R. Kaiser (2008). Alternatives for Scheduling Virtual Machines in Real-Time Embedded Systems, Proc. of the 1st Workshop on Isolation and Integration in Embedded Systems, Glasgow, Scotland, UK, pp.5-10.
[13] Rik van Riel (2015). Real-time KVM from the ground up, KVM Forum 2015, Red Hat.
[14] J. Kiszka (2009). Towards Linux as a Real-Time Hypervisor, Proc. of the 11th Real-Time Linux Workshop, Dresden, Germany, pp.205-215.
[15] W. Jiang, Y.S, Zhou, Y. Cui, W. Feng, Y. Chen, Y.C, Shi and Q.B, Wu (2009). CFS Optimizations to KVM Threads on Multi-Core Environment, Proc. Of the 15th International Conf. on Parallel and Distributed Systems, Shenzhen, China, pp.348-354.
[16] M. Rosenblum and T. Gar¯nkel (2005). Virtual Machine Monitors: Current Technology and Future Trends, Computer, vol.38, pp.39-47.
[17] Ruhui Ma, Fanfu Zhou, Haibing Guan (2013). Performance Tuning Towards a KVM-based Embedded Real-Time Virtualization System. J. Inf. Sci. Eng., vol .29, 1021-1035.
[18] Hyoseung Kim, Shige Wang, R. Rajkumar (2015). Responsive and Enforced Interrupt Handling for Real-Time System Virtualization. 2015 IEEE 21st International Conference on Embedded and Real-Time Computing Systems and Applications, 90-99.
[19] S. Brosky and S. Rotolo (2003). Shielded processors: Guaranteeing sub-millisecond response in standard Linux, In Workshop on Parallel and Distributed RealTime Systems, WPDRTS'03, Nice,France.
[20] Guang Xiaoming, Hujie, Chen Long, Guo Jing (2012). Principle and Implementation of Virtualization Technology, Electronic Industry Press.
[21] The relationship between virtual machine network card and tap device on linux bridge, https://blog.csdn.net/xiakewudi/article/details/76851076 April 30, 2021.
[22] RealTime Linux Wiki, 2009.http://rt.wiki.kernel.org.
[23] Steven Rostedt, Darren V. Hart (2007). Internals of the RT Patch. 2007 Linux Symposium, Volume 2. 161-172. Canada.