Barriers to Real-Time Network I/O Virtualization: Observations on a Legacy Hypervisor

Sang-Hun Lee  Jong-Soo Seok  Hyun-Wook Jin

Department of Computer Science and Engineering, Konkuk University
Seoul, Republic of Korea
E-mail: {mir1004,bluz,jinh}@konkuk.ac.kr

Abstract

Virtualization is considered as one of promising technologies to provide an efficient run-time environment for real-time embedded systems with respect to easy consolidation and safety. However, there are still several issues have to be addressed for real-time network I/O virtualization. In this paper, we briefly discuss about the clock synchronization, the jitter of I/O latency, and the tradeoff between latency and bandwidth.

Keywords: Real-Time, Virtualization, Network I/O, VirtualBox.

1. Introduction

As the complexity of real-time embedded software increases drastically, providing an efficient run-time environment especially in terms of consolidation and safety is becoming more important. The virtualization technology is considered as an authentic solution to provide an isolated run-time environment for real-time applications. Thus, there have been several researches to extend existing hypervisors to consider real-time constraints [1-2]. However, due to the high overheads of legacy hypervisors, it is not easy to exploit widely these on various embedded systems. To overcome such limitations, lightweight hypervisors originally targeting embedded systems are recently introduced [3-4]. However, we believe that there are still several additional issues have to be addressed to provide a “real” real-time network I/O virtualization. In this paper, we try to briefly discuss some of those issues.

2. Observations on a Legacy Hypervisor

There should be many design issues for real-time hypervisor in terms of network I/O, but in this paper, we selectively discuss three issues: i) clock synchronization, ii) jitter of I/O latency and iii) tradeoff between latency and bandwidth. Our discussions are based on observations with VirtualBox [5], an open-source type-II full virtualization hypervisor.

Clock Synchronization: In real-time embedded systems, the clock synchronization between guest domains is highly desirable for various application domains [6]. However, it is very tricky in virtualized environments because the hypervisor provides a virtual timer to the guest domain and keeps adjusting it based on the real (i.e., physical) timer. For example, VirtualBox inspects the virtual timer every 10 seconds and tries to adjust it gradually by changing the speed of the timer. That is, the virtual timer is basically not accurate, and its speed keeps changing. Such inaccuracy results in discrepancy between virtual timers in different guest domains. In order to mitigate the inaccuracy, we apply IEEE 1588, a time synchronization protocol, to guest domains running on the same node and measure its performance. Figure 1(a) shows the time differences between two guest domains, but the results still show the time difference larger than 100ms even with IEEE 1588.
Jitter of I/O Latency: The hypervisor (or back-end driver) handles many kinds of I/O events generated by the guest domains. The I/O event handling is hardly parallelized until these events are finally passed to the physical I/O devices. This increases I/O overheads but also induces significant jitters of I/O latency. For example, Figure 1(b) shows cumulative probability of virtualized Controller Area Network (CAN) I/O latency [7]. As we can see, the virtualized CAN has a very high jitter. The VirtualBox creates various threads for each virtual machine to handle different classes of I/O events. These threads compete against each other for processor resources and dominantly cause the jitter of I/O latency.

Tradeoff between Latency and Bandwidth: Real-time networks, such as Audio Video Bridging (AVB) [8], estimates the worst case latency between two end nodes, which is utilized importantly in providing QoS. However, the worst case latency can be under or over estimated in virtualized environments because of the I/O buffering of hypervisor. For example, the virtio [9] implemented in VirtualBox checks transmission requests of guest domains every 250us. Thus, in the worst case, a message of a guest domain can be delayed up to 250us. Figure 1(c) shows the increased latency due to the I/O buffering. However, we cannot blindly remove the buffering for low latency because this degrades the network bandwidth.

3. Conclusion

In this paper, we have briefly observed three issues; clock synchronization, jitter of I/O latency and tradeoff between latency and bandwidth. As on-going work, we are trying to overcome these issues.

![Figure 1](image-url) (a) Time difference between guest domains, (b) Cumulative probability of CAN latency, (c) Impact of buffering on TCP/IP latency

4. Acknowledgement

This work was supported by the Dual Use Technology Program (UM13018RD1).

References