The Journal of Systems and Software 85 (2012) 2593–2603

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Optimizing virtual machines using hybrid virtualization

Qian Lin [a], Zhengwei Qi [a,*], Jiewei Wu [a], Yaozu Dong [b], Haibing Guan [a]

[a] Shanghai Key Laboratory of Scalable Computing and Systems Shanghai Jiao Tong University, Shanghai, PR China
[b] Intel Open Source Technology Center, PR China

ARTICLE INFO

Article history:
Received 10 October 2011
Received in revised form 31 May 2012
Accepted 31 May 2012
Available online 9 June 2012

Keywords:
Hybrid virtualization
Hardware-assisted virtualization
Paravirtualization

ABSTRACT

Minimizing virtualization overhead and improving the reliability of virtual machines are challenging when establishing virtual machine cluster. Paravirtualization and hardware-assisted virtualization are two mainstream solutions for modern system virtualization. Hardware-assisted virtualization is superior in CPU and memory virtualization and becoming the leading solution, yet paravirtualization is still valuable in some aspects as it is capable of shortening the disposal path of I/O virtualization. Thus we propose the hybrid virtualization which runs the paravirtualized guest in the hardware-assisted virtual machine container to take advantage of both. Experiment results indicate that our hybrid solution outweighs origin paravirtualization by nearly 30% in memory intensive test and 50% in microbenchmarks. Meanwhile, compared with the origin hardware-assisted virtual machine, hybrid guest owns over 16% improvement in I/O intensive workloads.

1. Introduction

System virtualization is becoming ubiquitous in contemporary datacenter. Consolidating physical server by building virtual machine cluster is universally adopted to maximize the utilization of hardware resources for computing. Two fundamental but challenging requirements are to minimize virtualization overhead (Mergen et al., 2006) and to guarantee the reliability building virtualized infrastructure. Therefore, low level design of VM architecture is of great significance.

The conventional x86 architecture is incapable of classical trap-and-emulate virtualization, causing paravirtualization to be the optimal virtualization strategy in the past (Barham et al., 2003; Adams and Agesen, 2006). Recently, hardware-assisted virtualization on x86 architecture has become a competitive alternative method. Yet Adams and Agesen (2006) compared the performance between software-only VMM and hardware-assisted VMM, and the statistics showed that HVM suffered from much higher overhead than PVM owing to the frequent context switching, which had to perform an extra host/guest round trip in the early HVM solution. However, the latest hardware-assisted virtualization improvement introduces heavy overhead. Hardware-assisted paging (Neiger et al., 2006) allows hardware to handle the guest MMU operation and translate guest physical address to real machine address dynamically, accelerating memory relevant operations and improving overall performance of the HVM.

Although hardware-assisted virtualization performs well with CPU intensive workloads, it manifests low efficiency when processing I/O events. Our experiment shows that PVM performs up to 20% lower CPU utilization than HVM with the 10 Gbps network workload. The interrupt controller of HVM originates in the native environment with fast memory-mapped I/O access but is suboptimal in the virtual environment due to the requirement of trap-and-emulate. Frequent interrupts lead to frequent context switches and high round trip penalty, particularly for multiple virtual machines (Menon et al., 2005).

Consequently, hardware-assisted virtualization is superior in CPU and memory virtualization, and software-only virtualization owns optimized features for I/O virtualization. In practice, the performance issue is very workload-dependent because most real world applications are the mix of CPU and I/O intensive tasks. Therefore, hybrid virtualization techniques (Adams and Agesen, 2006) become promising. Nevertheless, the previous Hybrid VMM prototype (Adams and Agesen, 2006) leveraged the guest behavior-driven heuristics to improve performance. But its performance gain heavily depended on the prediction accuracy, and became marginal for modern workloads.

The contribution of this paper is a practical one. We propose a novel hybrid solution which takes both superiority features of PVM and HVM, and implement the prototype on Xen platform. The principal idea of our hybrid virtualization is to run the paravirtualized guest in the HVM container to reach maximum optimization. The Hybrid VM primarily features less MMU operation latency benefited from hardware-assisted paging technique and lower interrupt disposal overhead profited from the paravirtualized event channel. Besides, the original hardware-assisted virtualization environment

* Corresponding author. Tel.: (+86) 021 34205595.
E-mail address: qizhwei@sjtu.edu.cn (Z. Qi).

0164-1212/$ – see front matter © 2012 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.05.093

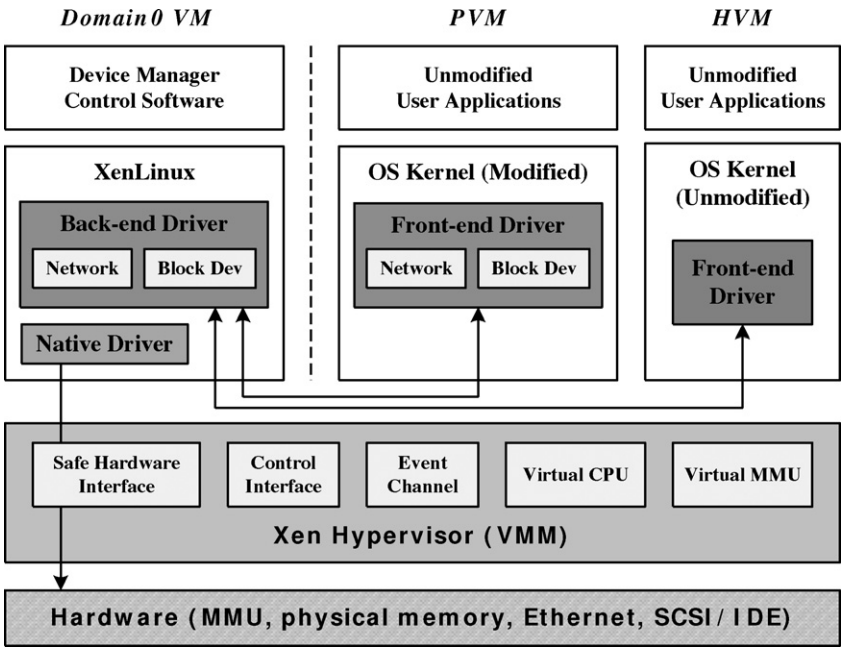

Fig. 1. Xen architecture. The Xen hypervisor is managing three types of VM. Domain0 plays an administrator role and supplies service for DomainU involving PVM and HVM. The front-end device drivers in DomainU communicate with the back-end drivers in Domain0 through device channel.

suffers from the issue of timer synchronization, which renders different timer resources hard to keep their relative timing pace to guarantee the timing correctness of VMs. We propose a feasible solution within the hybrid virtualization to solve this problem.

The rest of this paper is organized as follows. Section 2 introduces the background of virtualization as well as the software and hardware approaches with their advantages and disadvantages. Section 3 presents the hybrid virtualization architecture and design details. Section 4 deeply analysis the key factor affecting the efficiency of system call in guest OS, which can be treated as the performance indicator of VM. Section 5 specifically discusses the issue of timer synchronization under virtualized environment and the solution accessed by the hybrid virtualization. Section 6 analyzes the performance evaluation to demonstrate the performance improvements of the hybrid virtualization. Section 7 summarizes related work and Section 8 concludes.

2. Pros and cons in different virtualization mechanisms

With the promotion of virtualization technology, software-only and hardware-assisted virtualization approaches play different superiority in various fields. In this section, we firstly introduce the background of two mainstream virtualization techniques, and then present the details about advantages and disadvantages between them.

2.1. Paravirtualization

Xen (Barham et al., 2003; Clark et al., 2004) is famous for supporting paravirtualization (Whitaker et al., 2002). The Xen hypervisor locates between the physical hardware layer and the guest OS layer, as shown in Fig. 1 (Liu et al., 2006). The Xen hypervisor runs at the lowest level and owns the most privileged access to hardware. Among various VMs, Domain0 plays an administrator role and provides service for DomainU VMs. Domain0 also extends part of the functionalities of hypervisor. For example, Domain0 hosts back-end device drivers to manage the device multi-access from VMs, which utilize front-end device drivers and device channel to communicate with the back-end foundation (Xen.org, 2008).

The PVM guest kernel requires purposive modifications to adapt efficient software-only virtualization (Barham et al., 2003). Generally, x86 CPU privilege level is distinguished by different rings, where Ring0 is most privileged and Ring3 is least. As the hypervisor requires higher privilege level than VMs, the PVM guest kernel yields Ring0 to the hypervisor. Since paravirtualization does not change the application interfaces, user software can run in the Xen environment without any modification. Besides, paravirtualization uses DPT (Barham et al., 2003) as its memory virtualization strategy. In order to avoid page table switch at the time of hypervisor/guest boundary crossing, DPT modifies guest page table to be suitable for hardware processor usage as well as guest OS access. By modifying guest kernel, DPT partitions address space between guest OS and hypervisor, utilizing segment limit check to protect hypervisor from guest access. It reserves certain area of address space from each guest kernel to be dedicated to hypervisor usage. Consequently, each PVM shares its page table with the hypervisor so that the hypervisor can paravirtualize the guest paging mechanism.

2.2. Hardware-assisted virtualization

Hardware-assisted virtualization technique simplifies the design of virtualization management layer, i.e. hypervisor, and enhances the general performance with the help of processor virtualization. The conventional virtualization technique of *dynamical binary translation* was a compromising solution for system virtualizaiton without guest OS modification. The critical issue of dynamical binary translation is its low performance efficiency and design complexity due to the incapability of classical trap-and-emulate virtualization with previous generation of x86 architecture. Nevertheless, modern x86 architecture with hardware-assisted virtualization extension has fixed the trap-and-emulate virtualization hole on the architecture level, which highly reduces the design complexity of hypervisor. Hardware-assisted virtualization becomes an alternative and improved solution replacing dynamical binary translation. Furthermore,